Mobile Agents and Mobile Devices: Friendship or Difficult Relationship?

Oscar Urra, Sergio Ilarri, Raquel Trillo and Eduardo Mena

***Abstract*—Mobile agent technology has traditionally been recognized as a very useful approach to build applications for mobile computing and wireless environments. However, only a few studies report practical experiences with mobile agents in a mobile medium. This leads us to the following question: can current mobile agent platforms be used effectively in environments with mobile devices?**

In this paper, we study existing mobile agent platforms by analyzing if they are suitable or not in a mobile environment. We identify some key missing features in the platforms and highlight the requirements and challenges that lie ahead. With this work, we expose existing problems and hope to motivate further research in the area.

***Index Terms*—Mobile agents, mobile devices, mobile agent platforms, wireless and mobile environments.**

I. Introduction

MOBILE agents [9] are programs that execute in contexts called *places*, hosted on computers, and can autonomously travel from *place* to *place* resuming their execution there. Thus, they are not bound to the computer where they were created and they can move freely among computers.

Mobile agents provide interesting features, thanks to their autonomy, adaptability, and capability to move to remote computers. Thus they can carry the computation wherever it is necessary, without the need of installing specialized servers there (only a generic *mobile agent platform* [18] is needed). In particular, the interest of mobile agent technology for wireless environments has been emphasized in the literature (e.g., see [16]). Thus, for example, instead of communicating a large amount of data from a computer to a mobile device, a mobile agent can move to the computer where the data are stored to process the data locally, filtering the non-relevant data that should not be communicated through the network. As another example, a mobile device could use a mobile agent to perform a processing-intensive task on a fixed computer with the required resources, relieving the overload of the mobile device. This will increase, in turn, its battery life, an important limitation on these devices.

However, and despite there are many research works that emphasize the advantages of mobile agent technology in the context of distributed systems, there are only a few reported experiences on the use of mobile agents in real wireless environments with mobile devices. In most cases, a simple and static wireless environment is considered (e.g., we performed an experimental evaluation in this context in [20]). Moreover, we also believe that it is important to study the challenges that mobile environments imply for a mobile agent platform. For example, in ad hoc networks multi-hop protocols may be required for an agent to travel to a certain node (as each node can only communicate with other nodes within its communication range). As another example, it becomes apparent that mobile agent platforms should provide services for agents to discover other nodes.

This work studies in depth the requirements and current limitations of mobile agent platforms to be usable in a mobile environment, extending the study presented in [21]. We analyze the challenges that need to be solved and highlight the advantages and disadvantages of existing platforms in a mobile environment. The structure of the rest of this paper is as follows. First, in Section II we overview the technological context of this work. In Section III, we introduce the concept of mobile agent platform and motivate the development of this work. In Section IV, we study security issues. In Section V, we consider the challenges of using wireless communications and mobile ad hoc networks. In Section VI, we focus on architectural elements that must be considered. In Section VII, we describe some agent-based system's development issues. In Section VIII, we present some sample application scenarios. Finally, in Section IX we summarize some conclusions.

II. Technological Context

Having mobile agents working on mobile devices involves a set of technological components that should provide the expected functionality when all of them work together. This section describes the most important aspects of wireless communications, mobile devices and operating systems for mobile devices, and mobile agent platforms.

A. Wireless Communications

Mobile devices communicate by using radio signals. There are many standards and protocols used for this purpose. We will focus on the two most commonly used today for local wireless technologies:

- *Bluetooth* [11] was designed for small devices, such as cell phones and PDAs, and it is normally used for the transmission of small amounts of data or to connect to nearby compatible peripherals (e.g., printers, keyboards, or hands-free headsets). It has a limited bandwidth (maximum 3 Mbps), a range of up to 100 meters, and low power consumption.

Oscar Urra, Sergio Ilarri, Raquel Trillo and Eduardo Mena are with the Department of Computer Science and Systems Engineering, University of Zaragoza
E-mail: ourra@ita.es, silarri@unizar.es, raqueltl@unizar.es, emena@unizar.es

- *Wi-Fi* [13] networks, based on the IEEE 802.11 standards, allow to expand traditional Ethernet local area networks to places where either cabling is not an option or mobility is desired or needed. Its popularity is growing very fast and almost every laptop computer made in the last years has a Wi-Fi interface which allows it to connect to an also increasing number of public access networks in places such as hotels, airports and restaurants. Compared to bluetooth, it has a higher bandwidth (54 Mbps), a similar range, and a higher power consumption and cost.

Both bluetooth and Wi-Fi are freely available and they are very popular. The main drawback of these short-range technologies when compared to traditional fixed Ethernet networks is that the latency and bandwidth are not only worse, but also variable depending on the distance and obstacles existing between the sender and the receiver. Moreover, even an established communication could suddenly end if one of the devices moves out of its communication range or enters a shadow area such as a tunnel or an underground room (disconnections are frequent).

B. Mobile Devices

Mobile devices could be classified in three basic types:

- *Cell phones*. They are small, light, cheap and with little computation capabilities. Data communications can be carried out through mobile phone networks or via bluetooth. Mobile phone networks have a variable bandwidth depending on the transmission technology (GSM, GPRS, UMTS, etc.) and are available almost everywhere, but using these networks has an economic cost.
- *PDAs or pocket computers*. They are bigger and more expensive than cell phones, but they also have better processing capabilities. There are many architectures (ARM, MIPS, Xscale, etc.) and several operating systems (see Section II-C) that allow the execution of end user applications similar to those available in desktop computers. Communications can be established through bluetooth, and more recently also via Wi-Fi.
- *Laptop computers*. They have capabilities comparable to those of desktop computers. They usually have both Ethernet and Wi-Fi interfaces, but it is also possible to use bluetooth for data interchange with small devices.

Currently, the three types of mobile devices mentioned above are starting to mix. Thus, cell phones and PDAs are converging into a single device, called *Smartphone* (PDA with a SIM card), being the Apple iPhone one of the most popular Smartphones nowadays. Similarly, laptops and PDAs are mixing into the so-called *Netbooks*, which have less computing capabilities than a conventional PC but also have a lower cost. Another advantage of Netbooks is that most of them (like the Asus Eee PC, or the Acer Aspire One) have the same x86 architecture than PCs and can use the same operating systems and applications.

C. Operating Systems on Mobile Devices

The operating system is the software that act as an abstraction layer between the hardware and the user applications, allowing the programmer to access the different components of the device in a uniform way. In the case of mobile devices, which have a number of special features, it is even more important that the operating system be able to manage the resources in an efficient and flexible way. For example, the duration of the battery (which is a critical factor), the user interface (the keyboard or touch screen), the size and occupation of the internal memory, etc., are examples of resources that need to be considered.

There exist many operating systems that can run on mobile devices, but the most extended ones are the following:

- *Symbian*. It was created for its use on mobile phones by a consortium of manufacturers such as Nokia, Motorola, Samsung or Sony-Ericsson. It is available only for the ARM architecture and is bundled with high-end phones produced by these manufacturers. It is proprietary but expected to become open source during 2009.
- *Windows Mobile*. It was developed by Microsoft and is used by many manufacturers (some of them using other systems too) such as Hewlett-Packard, Samsung, Motorola or Qtek. It is available for the ARM, MIPS and x86 architectures, and there exist two variants: *Windows Mobile PocketPC* for PDAs and *Windows Mobile Smartphone* for high-end mobile phones.
- *Android*. It is the most recent operating system that has appeared in the mobile market, developed by Google. It is derived from Linux and it is licensed also as open source. It is available for the ARM, MIPS and x86 architectures and can run on mobile phones, PDAs, and PC laptops. Due to its novelty, only a few devices use it nowadays, produced by manufacturers such as HTC or Samsung.

The operating system is a key element that should allow an efficient exploitation of the device's resources. Different features of an operating system, such as its scheduling policy, may have an important impact on the performance of a mobile agent platform running on the device.

D. Mobile Agent Platforms

There are many mobile agent platforms available, that differ in several aspects (such as their general architecture, communication style, etc.) and compare differently in terms of performance, reliability or scalability [18]. However, all of them offer similar services to their agents: execution, communication, mobility, tracking, directory, persistence, security, etc. We will briefly describe in this section the main services.

All platforms provide an *execution environment*, which is the most basic service and allows agents to run their code and access other services offered by the platform. The use of bytecode-based languages like Java eases the implementation of this service and its portability between different hardware architectures, and therefore most platforms are implemented in Java. Every device can execute one or more instances of these environments (known as *places* or *containers*), where several agents can be running simultaneously.

One of the strongest points of agents is their communication capability, so the existence of a *communication service* is also very important. When an agent starts a communication, it needs to determine the message to be transmitted and the destination where it must be sent (i.e., the target agent/s). The message must contain the information to transmit and must be intelligible for both the sender and the receiver. To achieve such mutual comprehension, there exist different *agent communication languages* (ACLs) that make the communication among different agents possible [4], [5]. Regarding the destination of the message, it can be a single agent or many of them, which can be located in the same execution environment as the sender or in a remote location. The communication service should provide agents with a common mechanism to build and deliver their messages to their destination, independently of where they are located (location transparency).

The *mobility service* allows agents to move to other execution environments. The process involves three steps. First, the agent determines the destination place and invokes the mobility service to be transferred. Then, its code and data are transferred across a network connection to the destination, where another running platform receives them. Once the transmission has finished without errors, the copy of the agent in the origin is destroyed and a new one is created in the destination from the code and data that compose the agent. This process is fail-proof: If there is any problem with the trip of the agent, the agent that attempted to travel will get the control back and decide what to do next.

The *object tracking service* keeps a record of the location of all the objects present in a multi-agent system [15], such as the agents themselves or the places/containers available in the distributed system. Whenever a new object is created, destroyed, or moved in the system, the service must be aware of such an action and update its location. As mentioned before, it is very important for the agent programmer that this service provide a truly *location transparency* so that, once an object's reference is obtained, it will be kept up-to-date by the system as long as necessary (the programmer will not need to refresh/update this reference).

Besides the services offered by the platform, agents can be programmed to offer different services to other agents or software components. Similarly, agents may also need to use services offered by other agents, to achieve their goals. The *directory service* allows the registration by agents of a description of the services they provide, and a common way to query, locate, and access the services included in the registry.

Thanks to all these services, mobile agents can *live* in a generic distributed environment and perform their tasks effectively. In the next section, we will examine some special difficulties introduced by mobile environments. A mobile agent platform should take this difficulties into account in order to be usable in such an environment.

III. Mobile Agent Platforms in Wireless Environments

Mobile agent technology has been proposed as a key element in the development of many applications, for a variety of reasons (e.g., their capability to exploit the locality of data by moving to the data source instead of interacting remotely using a network). With the increasing popularity of mobile devices, it was a natural step forward to try to apply mobile agents to the mobile environment. Given its portable nature, mobile devices use wireless communications, creating a scenario completely different from 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 a short communication range based on wireless technologies –that usually offer a low bandwidth, a high latency, and intermittent/unreliable connectivity–. Thanks to their features, mobile agents can be very useful in wireless applications (e.g., see [16]), as they could help to reduce the negative effects of such limitations. For example, as mentioned before, mobile agents can move to the place where the information is stored and process the data locally, discarding the data that is not relevant and therefore needs 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 evaluating alternatives continuously based on information about their environment.

However, despite the increasing popularity of wireless services and the advantages of mobile agents in these environments, there are not many practical applications of this technology, except for some proofs of concept. A possible explanation could be that it is very difficult to develop and maintain such mobile agent based applications because existing platforms lack a number of features that should be present for their use in a wireless and mobile environment. Among them, we could emphasize:

- Features related to security, since the use of wireless communications broadcast data that could be intercepted or altered without having any notice.
- Features related to special network topologies, since there may be multiple mobile nodes with short range and unstable communications, which can make the process of transferring data between two nodes challenging.
- Features related to the way the platform itself works, since the mobile agents need different services, such as transportation or communication services.
- Features that should help the developer of agent-based systems, such as monitoring or debugging tools.

Providing the features listed above is important to develop applications based on mobile agents for mobile environments. Thus, for example, an agent should be able to detect the availability of new nearby devices (e.g., to travel to them) and to communicate with other agents easily and efficiently. Mobile agent platforms have usually been developed with a static context in mind and now they must be adapted to a more open and dynamic environment.

Besides, scalability and reliability are two key features which are important even when the mobile agents rely on fixed networks, and therefore even more critical in a challenging environment with wireless networks. Some tests indicate that the scalability/reliability of some platforms should be improved [3]. However, as this need is not tied to the wireless case –although heightened by it– we will focus on the other features; a summary of the analysis performed in this paper is presented in Table I. In the rest of the paper we will consider the following platforms: JADE, Voyager, and SPRINGS.

*J*ADE (http://jade.tilab.com), developed by Telecom Italia Lab since July 1998, was released as open source in February 2000 (last version: JADE 3.8.1, November 2008). It is a popular FIPA-compliant agent platform. An agent is composed of different concurrent (and non-preemptive) behaviors, which can be added dynamically. Among the benefits, we could indicate that there is a wide variety of tools provided (e.g., for remote management and monitoring of agents, and to track interchanged messages) and it can be integrated with different software such as Jess (a rule engine which allows agents to reason using knowledge provided in the form of declarative rules). Besides, it supports the development of ontologies to represent the knowledge of agents.

*V*oyager (http://www.recursionsw.com), developed initially by ObjectSpace in 1997 and currently by Recursion Software (last version: Voyager Edge 7.2.0, April 2009). It is a distributed computing middleware focused on simplifying the management of remote communications of traditional CORBA and RMI protocols. It is a commercial product, but there exists a *Community Edition* freely available valid during one year.

*S*PRINGS (http://sid.cps.unizar.es/SPRINGS/) [3], developed by the Distributed Information Systems Group at the University of Zaragoza in Spain, focuses on scalability and reliability in scenarios with a moderate and high number of mobile agents. Different features of other popular platforms, such as Voyager, have inspired its development.

IV. Security Issues

Security is always a major concern. Besides security problems with mobile agents in fixed networks [1], [22], other problems particular to wireless environments arise. Thus, it is necessary to ensure the privacy of the data and its integrity, and consider authentication and trust issues.

A. Communication Encryption

Wireless communications usually broadcast the transmitted data in an omnidirectional way. The disadvantage is that these data will be received not only by the intended destination but also by anyone within the range of the originating communication device. Although almost every wireless communication protocol (such as Wi-Fi or Bluetooth) can encrypt the data, it is not mandatory and in some circumstances an unencrypted communication can be the only form available (e.g., with public access points or *hotspots*). If a mobile agent is transferred using an insecure channel, its code and data will be exposed to any nearby device. To avoid this problem, the mobile agent platform should be able to encrypt all its communications when connecting to others.

Regarding the considered existing mobile agent platforms, both Voyager and JADE-LEAP can natively use SSL connections –which assure data flow encryption–, whereas SPRINGS currently lacks this feature.

B. Code Integrity

The code of a mobile agent can be altered, either intentionally or accidentally, in many ways. For example, in the first case, a malicious user could modify it while it is running on its device. In the second case, a failure such as a memory loss or a problem with the wireless transmission of the agent could lead to a corruption of its code. The execution of an agent whose code has been altered can lead to unpredictable consequences and should be avoided. Thus, the mobile agent platform should verify the agent's code integrity before starting its execution, for example signing the code with a X.509 digital certificate.

Thanks to the use of SSL by Voyager and JADE-LEAP, any attempt to tamper with an agent while it is being transmitted will be detected. Moreover, Voyager provides a signing mechanism to assure that the agent is not modified while it is stored in the device's memory. The current version of SPRINGS does not provide these features.

C. Authentication and Trust

A mobile device has a number of resources (such as the CPU, the memory, the file system, the user interface, etc.) susceptible to be used by an agent to accomplish its tasks. To avoid abuses on the use of these resources, the platform should state the extent to which it trusts an incoming agent –this is critical in an open environment–. Depending on it, the platform should allow, limit or even deny the mobile agent the access to the resources. There already exist mechanisms of authentication for distributed environments (e.g., Kerberos) that could be used to determine the owner of an agent. Once it is authenticated, different access control policies can determine the resources that the agent can use.

Voyager and JADE-LEAP have mechanisms to verify the identity of agents and other platform components, and grant different privilege levels. SPRINGS currently offers no authentication mechanisms but a basic access control.

V. Network Issues

Existing mobile agent platforms assume the existence of a TCP/IP network where every node can potentially connect to any other. This approach has the advantage of its simplicity, but it prevents considering explicitly other forms of communication that could be beneficial in a context with wireless mobile agents.

One limitation of mobile devices is that their communication interfaces (usually Wi-Fi or Bluetooth) have a relatively short range (a few hundred meters). Therefore, in a mobile ad hoc network a multi-hop routing protocol may be needed for two nodes to communicate. For a mobile agent platform, it would be interesting to access information about: whether the connection is fixed or wireless, the bandwidth available, if an established link (e.g., a Wi-Fi connection) is encrypted or not, the coverage level or the strength of the received signal in the case of a wireless communication, the identifiers of nearby wireless networks or nodes that could be contacted, etc.

With all this information, the platform would be able to take different useful decisions. For example, it could select the most appropriate communication link if there are several

TABLE I
COMPARISON OF MOBILE AGENT PLATFORMS: FEATURES FOR MOBILE ENVIRONMENTS

Feature		**Voyager**	**JADE/LEAP**	**SPRINGS**
Security Issues	Encryption	Yes (SSL)	Yes (SSL)	No
	Code integrity for transmissions	Yes (SSL)	Yes (SSL)	No
	Code integrity for execution	Some (signing)	No	No
	Authentication and trust	Yes (very rich)	Yes (rich)	Very basic
Network Issues	Access to link layer	No	No	No
	Adaptability	Latency measure	Conn. status	Conn. retrying
Architectural Issues	Discovery of nodes	No	No	No
	Transparent service discovery	No (YP)	No (DF)	No
	Adapted tracking approach	No (AgentSpaces)	No (AMS)	No (RNS)
Programming Issues	JVM supported	IBM J9	IBM J9	IBM J9
	Network communications	Sockets, HTTP	Sockets, HTTP	RMI
	Graphical interface	No	Yes	No
Strong points		Messaging, devices	Messaging, ontologies	Scalable, reliable

options available, in the case of a secure channel it would avoid encrypting the communicated data –reducing the CPU utilization and therefore saving battery power in the mobile device–, it could decide not to retry a failed communication if the coverage is poor, etc. Making this information accessible to the agents would also be interesting. For example, an agent could decide to jump to another device/node if its current device is getting out of coverage.

As far as we know, current mobile agent platforms do not have the ability to obtain this kind of information, which would be important in order to make the platform truly adaptive to different network environments. The closest related features that the considered platforms have are the following: Voyager can measure the network latency and detect that a communication is broken when an abnormally high value is obtained; JADE-LEAP can get basic information about the link status (connected or disconnected); finally, SPRINGS retries failed communications automatically according to some defined policy.

VI. Architectural Issues

In this section, we provide an overview of some architectural issues that should be considered in a mobile agent platform suitable for a mobile environment: discovery services and tracking and name services.

A. Automatic Discovery Service

As opposed to a fixed distributed infrastructure for mobile agents, a mobile context usually presents an open environment where many different computers/devices may appear and disappear at any time. Therefore, it is of paramount importance to provide agents with appropriate mechanisms to locate nodes to where they can travel. Moreover, the capabilities/services offered by these nodes should be advertised to allow the agents to decide a convenient target node.

1) Discovery of Services: There exist several service discovery protocols, such as the *Service Location Protocol* (*SLP*), the *Universal Plug and Play* (*UPnP*), or the use of the *Jini* technology. The suitability of these protocols for ambient intelligence is analyzed in [14].

Mobile agents should be provided an automatic discovery mechanism to allow them the detection of services of interest[1]. However, as far as we know, these protocols have not been integrated in any existing mobile agent platform. Moreover, it is not clear if they are the best choice in a mobile environment. For example, JADE-LEAP has a federated Yellow Pages (YP) service, but allocating it to a node is not transparent to the programmer and movements of the mobile device may invalidate the convenience of using that allocated YP service.

Finally, we would like to highlight the importance of providing a *semantic matching* of services [17], not only *syntactic matching*, if we want to enable dynamic and flexible interactions among the mobile agents. Thus, *Agent Communication Languages* (*ACLs*) [4] could play an important role. The language proposed by the *Foundation for Intelligent Physical Agents* (the *FIPA* ACL –see http://www.fipa.org/–) is the most popular proposal and it is supported by several mobile agent platforms (e.g., JADE). If services for mobile agents are implemented as web services, as suggested in [23], then existing techniques for web services (UDDI, WSDL, SOAP, OWL-S, etc.) could also be adopted.

2) Discovery of Nodes: It is also necessary for an agent to be able to detect the nodes that are reachable and which services/features provide those nodes. Otherwise, it will be very difficult for the agent to assess the convenience of traveling to another node. Thus, nodes must be auto-descriptive. This is particularly important in a heterogeneous environment where there will be computers/devices with different processing and communication capabilities.

However, no platform provides mechanisms to allow an agent to detect other potential target computers/devices. Although some platforms provide name services to query the computers that can host an agent (e.g., the Region Name Server in SPRINGS [3]), they do not consider that some

[1]Some works propose using mobile agents to implement service discovery [8], [12].

computers/devices may not be accessible from a given location (e.g., if the mobile agent is executing on a device that can only communicate with other devices within its communication range); indeed, the name service itself may be unreachable.

B. Tracking of Mobile Agents and Name Services

As mobile agents move from one computer/device to another, tracking their locations efficiently (which is required if we want to be able to allow communications with those agents from any computer/device) is challenging. To solve this problem, different approaches have been considered. Some mobile agent platforms provide a naming service that can be used to locate an agent and then send a message to its address. An alternative (and also complementary) approach supports communications by using *proxies* (similar to the *stubs* in *RMI*) as a convenient abstraction to refer to remote agents (e.g., to send them messages or call methods remotely). Several platforms, such as SPRINGS and Voyager, support proxies. Related to the concept of proxies, *dynamic proxies* remain valid independently of the agents' migrations (i.e., the reference is updated automatically) [3].

However, existing strategies have been developed with a fixed network in mind and are not appropriate in a mobile environment. For example, SPRINGS assumes the existence of stable *Region Name Servers* (*RNSs*) and *location servers* with tracking responsibilities. This could be unsuitable in a dynamic context because nodes can leave/enter the network at any time and some nodes could be temporarily unreachable. Similarly, according to [10], Voyager uses forwarding chains of proxies, which may be inconvenient because any link (pointer) in the chain could disappear at any time. Thus, we believe that new tracking techniques are needed in this context, avoiding mechanisms that rely on the availability of certain nodes or centralized approaches, in favor of adaptive tracking approaches.

Finally, we should mention that ensuring the uniqueness of agent names and at the same time providing user-friendly mechanisms to address the agents is a challenge in an open and dynamic environment. A potential preliminary solution would imply a shift in the way agents communicate. At present, it is usually assumed that agents identify their partners by name. An alternative would be to identify partners by service, which would make user-friendly agent names unnecessary.

VII. Issues Related to the Development of Systems Based on Mobile Agents

The implementation of mobile agent applications for their execution in mobile devices can be problematic due to a number of issues, such as the availability of Java interpreters, the efficiency of different types of networking protocols for communication, or the existence of tools for the programmer.

A. Java Virtual Machines for Mobile Devices

To guarantee portability across heterogeneous devices and computers, Sun Microsystems created different distributions of the Java platform. From the point of view of this paper, it is interesting to distinguish the following possibilities:

- *Java SE* (*Java Standard Edition*). It is the most widespread edition, used by personal computers and servers running general purpose applications. There are many implementations and *Java Virtual Machines* (*JVMs*) available, usually for free, like the official implementation by Sun Microsystems for PC and SPARC architectures, as well as those offered by other companies such as IBM or HP.
- *Java ME* (*Java Micro Edition*). It is designed for small devices and there are two specifications, one intended for more limited equipment such as mobile phones (*Connected Limited Device Configuration -CLDC-*) and the other, more complete, for PDAs (*Connected Device Configuration -CDC-*).

Sun Microsystems does not develop JVMs for mobile devices, but they are made by third parties. In mobile phones, the manufacturers (e.g., Nokia, Motorola, Sony-Ericsson) include the runtime as an indivisible part of its device. This also happened with PDAs in some cases (e.g., with PDAs by Compaq), but now almost none brings any JVM installed, and so it must be acquired separately. JVMs for mobile devices are neither easy to find nor free, in contrast to the standard Sun's JVM. Among the virtual machines for PDA-like mobile devices, we can highlight the following:

- *Jeode*. Some years ago, it was distributed with PDAs by Compaq. Then, there was the fusion with HP, and the new models did not bring it. The company which originally made it was acquired by another one which no longer continued its development.
- *Cr-EME*[2]. It is a high quality J2ME-compliant commercial machine oriented to the market of embedded devices, and has some optional components such as RMI (see Section VII-B) or AWT, which are sold separately. Its biggest problem is its distribution, as it is only sold in lots of 40 units at 1000 dollars (as of July 2009).
- *IBM Websphere J9*[3]. Another commercial JVM. It is part of the Websphere development suite. It is also a very good option and also has additional components such as RMI and AWT. The price of the runtime (without the development environment) is 25 dollars (as of July 2009), which makes it quite affordable.
- *MySaifu*[4]. It is an open source implementation under development. It is capable of using some features of AWT components and user interfaces. However, it is still at a very early stage regarding networking aspects (e.g., it does not offer support for RMI).

Regarding the considered mobile agent platforms, all of them (SPRINGS, Voyager and LEAP) require the IBM Websphere J9 Java machine when they are executed on mobile devices like PDAs. Voyager, additionally, has a version using the *.NET Compact Framework*, a light version of the .NET by Microsoft.

[2]http://www.nsicom.com

[3]http://www-306.ibm.com/software/wireless/weme/features.html?S_CMP=wspace

[4]http://www2s.biglobe.ne.jp/~dat/java/project/jvm/index_en.html

B. Networking communication protocols

There exist different communication protocols that can be used for agent communication. The choice of one or another can be a difficult issue.

On the one hand, using low-level communication protocols gives a higher performance, but it is more difficult for the programmer. On the other hand, using a high-level protocol is easier, but less efficient and it could be unavailable on a mobile device with constrained capabilities.

Some popular network communication protocols are:

- *Plain sockets*. This is the most low-level communication form that can be used. It is standard, efficient, and it is possible to explicitly choose if they must be connection-oriented (TCP) or not connection-oriented (UDP). They can be used in almost any programming language but the drawback is that any higher-level operation must be programmed using many low-level functions, which is a very tedious task for the programmer and prone to fail.
- *RMI*. The *Remote Method Invocation* is the mechanism provided by Java to invoke functions on remote (or local) objects. The invocation can carry information through the use of input parameters, that can be as simple as integers or characters or as complex as whole objects. In some JVMs for mobile devices, it is an optional component and in others it is not available at all. Although RMI is a Java-only communication form, it fits well with the paradigm of mobile agent because most mobile agent platforms are implemented in Java.
- *HTTP*. The *Hyper-Text Transfer Protocol* used by the Internet browsers have many advantages: It is a standard and ubiquitous. It is also bundled with many programming languages and libraries, and can be easily programmed and debugged (e.g., using a browser). Additionally it can travel through proxies, giving the possibility of traversing firewalls, which are usually open for this protocol but closed for any other. A drawback of HTTP is that it can be inefficient to transfer binary data because in such case the data must be encoded (usually with MIME). Even though there are some proposals to implement mobile agent platforms using the HTTP protocol (e.g., see [7]), currently no popular mobile agent platform is implemented on top of it.

Regarding the considered mobile agent platforms, SPRINGS uses RMI, and both JADE/LEAP and Voyager use plain sockets for intra-platform communication and HTTP for extra-platform communication.

C. Programming and Debugging Tools

To ease the development of distributed applications using agent technology, a number of tools and utilities can be used by the programmer. These tools are not very different from those used for the development of any other software, but not all of them are so available or popular among existing mobile agent platforms. Some examples of interesting features for the developer follow:

- *Programming API and documentation*. In order to develop applications using mobile agents the programmer must be able to build the structure of agents, but he/she must also know the actions they can perform (i.e., move to another execution place, send a message to another agent, query a published service, etc.) and how to program the agent to perform these operations. The mobile agent platform's API allows the programmer to build agents and use from the most basic to the more complex features of the platform. This API should be complete enough to allow the programmer to use every platform capability, and have extensive documentation explaining its use, to ease the development and maintenance of such agent-based applications.
  Regarding the considered mobile agent platforms, all of them have a well-documented API, as well as additional documents that cover other interesting platform issues.
- *Deployment and management graphical tools*. A mobile agent platform can be a very complex software, with many components and configuration parameters. The process of setting up a scenario for the execution of a complex distributed application can be a tedious task, especially if the application development is in progress. The use of a graphical user interface can make more pleasant the task of setting up and deploying initially all the components of a multiagent system. Once it is running and all their components executing as expected, such a tool could also be useful for managing the whole environment, allowing to add or remove components at any time in an easy way.
  JADE/LEAP is the only platform that provides a graphical tool that allows to perform basic operations like creating places and agents, move agents from one place to another, and many others.
- *Logging capability*. During the development of any application, errors and bugs may appear in the software as a natural part of the process. Additionally, in a distributed scenario many components can be physically out of the scope of the programmers (i.e., in remote locations) so it can be difficult to know what is going on at every moment in such components. In a mobile agent platform these components can be agents, service directories, places, etc. It is very important that the different platform components have the ability to log all the relevant information (such as their status, executed actions, etc.) and store such data in a way that can be retrieved later by the programmer for their analysis.
  All the considered platforms give extensive information when some components (such as places or registries) are launched, that can be stored in plain text files. In general, the information provided consists of: the name and version of the platform, the name of the object, their correct or incorrect initialization, and important events and errors (service registration, the creation of an agent, etc.)
- *Debugging and monitoring capability*. Similarly to the logging capability, debugging/monitoring is also a desirable feature for the programmers. The difference with logging is that debugging/monitoring is performed *online*, so that the programmer can follow the status (data, ac-

tions, communications) of an object (agent, place, service, etc.) while it is being executed. Moreover, it is also very useful if the execution can be paused and analyzed, or even if it is possible to modify the contents of program variables or communications during the execution.

Neither Voyager nor SPRINGS provide natively this kind of tools, whereas JADE/LEAP has some debugging and monitoring functions in its graphical interface. Thanks to it, it is possible to follow agents as they move among places, stop their executions, send them customized messages, and even *sniff* and alter the messages the agents exchange. It is also worth mentioning the existence of independent monitoring tools for agent-based systems, such as the 3D monitoring tool presented in [2].

As a summary, in general the considered platforms provide enough tools to the programmer for developing agent-based applications properly although some advanced and handy tools are missing. One notable exception is the graphical interface of JADE, that allows the control and debugging of this platform in a very intuitive way.

The programming tools that we have described in this section are important in any scenario where a system based on mobile agents must be deployed. However, it is obviously in distributed scenarios, and particularly in mobile computing scenarios, where the programmer needs more support from the platform. Particularly, debugging and monitoring capabilities are very important in these contexts, where the number of potential sources of failure increases

VIII. Sample Application Scenarios

In this section, we will show two sample application scenarios that can benefit from mobile agents: The first is a distributed monitoring system using moving cars in a wide geographic area. The second is a simple system for searching documents on a set of personal PDAs.

A. Mobile Agent Technology on Vehicular Networks

A vehicular network (or VANET) is a mobile network whose nodes are cars moving across the roads and highways of a given geographic area. We think that mobile agents could provide interesting benefits in this scenario. As an example, we will summarize in the following our proposal to use mobile agents in cars for monitoring different environmental parameters without needing a fixed and costly infrastructure [19].

In this sample scenario, the cars are equipped with different types of sensors that measure the desired parameter as the cars move inside a designated area of interest or *Monitored Area*. The data collected by these sensors are carried by mobile agents hoping from car to car until they reach a *Monitoring Center* for later processing. In this way, many monitoring tasks can be performed simultaneously, covering a large area and without needing a fixed infrastructure (except for the Monitoring Center), although the mobile agents could also take advantage of already-existing fixed elements like roadside antennas or information panels. To perform the monitoring, all the cars participating in the monitoring VANET must have installed some technological elements such as: one or more sensors for reading the environment data, a wireless communication device (e.g., Wi-Fi or Ultrawide Band), a GPS receiver for knowing the position with respect to the monitored area, and a PDA or laptop capable of executing a mobile agent platform and accessing the rest of the elements. The whole monitoring process consists of five steps (see Figure 1):

1) The monitoring scope is defined. This implies setting the environmental parameters to measure, the location of the monitored area, and the amount of time during which the monitoring must be performed.
2) In the Monitoring Center a number of monitoring agents are created and they are transmitted (they *jump*) to moving cars (it is supposed that this Center has an antenna or is near the road where the monitoring cars are constantly moving).
3) The monitoring agents travel on the car towards the monitored area. At any moment, they can jump to any nearby car if it follows a more promising path towards the target area. To know this, the agents can read the GPS information and also ask other nearby agents about their position and heading.
4) Agents in the monitored area read data from their cars' sensors. They can also jump to other cars and clone themselves to stay within the area and increase the number of samples taken, thus improving the monitoring.
5) When the planned monitoring time expires, the agents must return the measured data to the Monitoring Center for later processing. To achieve this, they follow the same process that they followed to reach the monitored area, jumping from one car to another based on their position and other factors.

This application scenario shows that mobile agents in VANETs can be very useful for monitoring purposes. Moreover, we can think of other contexts where VANETs and mobile agents could fit together, for example to exchange information of the road status some kilometers ahead, to disseminate information in an area about traffic jams, or even to answer queries launched dynamically by drivers (e.g., how many parking sites are in a certain street).

However, using mobile agents in this highly dynamic mobile P2P environment also poses a number of interesting challenges due to the high mobility of the network nodes. Some of these challenges are closely related to those exposed in the previous sections: discovering services in a decentralized way, travelling from one place to another using a multi-hop protocol when nodes are constantly moving, or assuring that data is not altered during transmissions to avoid the dissemination of false information. We plan to extend our work on this area in order to study in detail the benefits and difficulties implied by the use of mobile agents in vehicular networks.

B. Searching Relevant Documents in a Set of PDAs

We may envision a situation where members of a research team in a university store different types of research documents (papers, technical reports, personal notes, etc.) on a wide range of mobile devices to be able to access these documents while they are moving. Besides the importance to keep synchronized

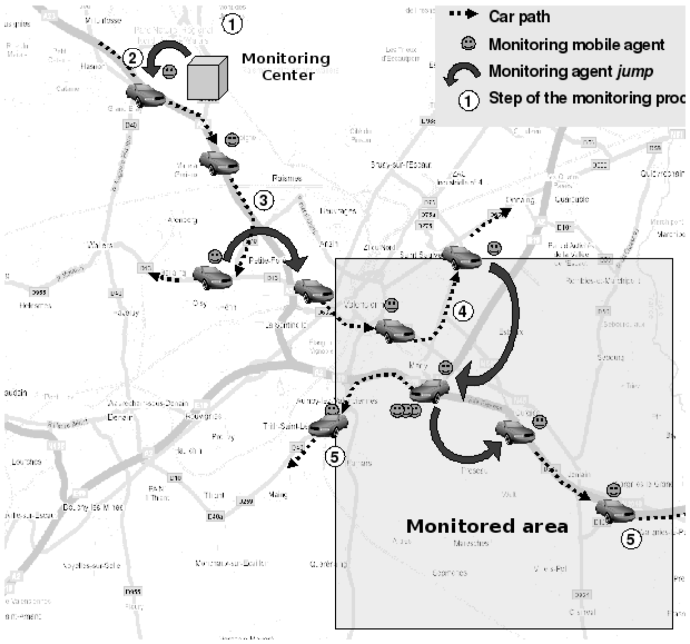

Fig. 1. Using monitoring agents for distributed measurement

the different repositories managed by a single person, it is also interesting to allow him/her to search for relevant documents that may be stored by other member of his/her team. We advocate the use of mobile agents for efficient searching and filtering of relevant documents. In this section, we will show an approach based on mobile agents and compare it experimentally with a traditional client/server approach. From any of these devices a search can be launched, in order to retrieve the most relevant files and transferring them wirelessly to a local directory. The purpose of this experiment is to verify that a distributed search can be more efficient by using mobile agents than using a traditional client-server method.

To start the searching process, the user launches an application in his/her own PDA and enters some keywords (a *search string*) to be searched in the documents. A simple algorithm is used to calculate the relevance of a file for a given search string. We will evaluate two alternative approaches to find relevant documents using that algorithm:

- *Using a client-server approach*. A file server runs on every device containing the files. The search is started from a client that connects to every other device, downloads its files and finally performs the search locally by analyzing all the documents downloaded from all the devices.
- *Using mobile agents*. Every device runs the SPRINGS mobile agents platform and one of them (the client) starts the search. A mobile agent is created and it moves to the other devices, one after another. In each of them, it makes a local search and the relevant files are sent to the origin by another mobile agent created then for that purpose.

The devices used for this test are one laptop and two PDAs. Bluetooth is used for wireless communication, acting the laptop as an intermediate access point for the PDAs. The configuration of the elements is as follows:

- The laptop computer is an Intel Centrino with two 1.66 GHz cores and 2 GB of RAM, running Linux with kernel 2.6.24. It acts as a bluetooth access point for wireless communications. The Sun JVM 1.4 is used, since in this way the same compiled code can run seamlessly in both the PC and the PDAs Java interpreters.
- A PDA Fujitsu-Siemens Loox 720, with a X-Scale processor at 520 MHz, and 128 MB of RAM.
- A PDA HP iPAQ 1940, with a Samsung S3C processor at 266 MHz, and 64 MB of RAM.
- Both PDAs run Windows Mobile 2003 and include an integrated bluetooth adapter (class 2, 10-meter range). The JVM used on the PDAs, in both cases, is the IBM Websphere J9 version 11 for Windows Mobile.

In the experiment, we measure the time that a search takes to complete, depending on the relevance of the search string in the available documents (e.g., a 20% relevance means that 20 out of 100 documents are relevant for the given search string). The parameters used for evaluation are as follows:

- The search is launched from the Fujitsu PDA.
- In the test using mobile agents, the SPRINGS' RNS component runs on the PC, as this is the most static of the three devices involved.
- In every device there are 10 files of 50 KB, adding up to a total of 500 KB.
- We vary the relevance between 0% and 100% in 20% steps, and for every value we repeat each test five times. Average values are reported in the experimental results.
- In every test, we measure the amount of time needed since the beginning of the search process until all the results are retrieved.

The results can be seen in Figure 2. The time needed to complete a search with the traditional client-server method is constant, regardless of the relevance of the search string. This is because, with the client/server approach, all the documents are communicated to the searching device independently of whether they are relevant or not for the search string considered. However, the amount of time needed with the approach based on mobile agents depends of the relevance (for the search string considered) of the different documents stored on the devices. As the figure shows, the use of mobile agents minimizes the amount of data transferred and leads to smaller search delays. Only if most of the documents are relevant, which implies that the local filtering performed by the agents is very limited, the client/server approach performs similarly.

There are two extreme points in the graph, for relevance values of 0% and 100%. In the first case the searching process takes ten seconds before the result (none) is obtained. This is due to the time it takes the agent to be dynamically created and because once it starts moving it makes all its journey through the PDAs sequentially, one site after another. In the 100%

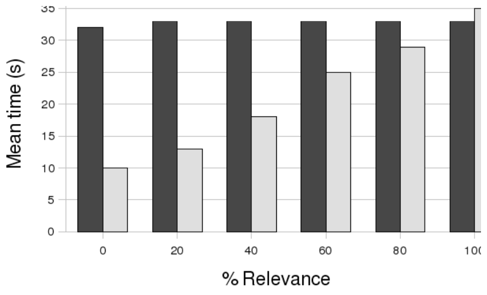

Fig. 2. Times searching with and without agents

case, the method using mobile agents is slightly worse than the traditional client-server approach, for the same reasons (extra overhead of mobile agents with no filtering benefit).

From this experiment we conclude that mobile agents can be used for efficient searching, as they process data locally on every data container, transferring only the relevant data. In this way, the workload is shared among all the devices and the bandwidth usage is minimized, making a better use of both resources, which are so limited on mobile devices.

An enhancement would be to use a more decentralized scenario, since in this case the PC is a central node containing the RNS and acting as a bluetooth access point. In a truly decentralized scenario the PDAs would communicate among them directly (without needing the PC) in a P2P way. Unfortunately, SPRINGS is not yet ready to use such decentralized solutions, and neither Voyager or JADE/LEAP. Another enhancement would be to use a higher number of PDAs and create many copies of the searching agent, that would execute their tasks in parallel. With these enhancements, the searching and filtering efficiency would increase.

In general, mobile agents can be useful in scenarios where we need to collect/disseminate some information from/to a set of mobile devices. For example, a similar scenario to the one presented here could be considered for schedule planning [6].

IX. Conclusions

Mobile agent technology has been highlighted as a very interesting approach to build applications for mobile environments. However, it is hard to find practical applications with real prototypes and using the available mobile agent platforms. One reason is probably that such platforms have been developed with a fixed distributed environment in mind, and not considering the features that may be of special interest in a mobile environment (e.g., reliance against security threats, adaptation to the network technology, and service/node discovery). Considering these features would make the adoption of the technology much easier.

In this paper, we have identified the requirements and desired features of mobile agent platforms to be used in scenarios with mobile devices. With these requirements in mind, we have analyzed the missing features in some popular mobile agent platforms. As future work, we plan to study these issues in detail and propose solutions for SPRINGS [3]. We hope that future research and development efforts will eventually lead to consolidate a good relationship between mobile agents and mobile devices.

Acknowledgment

This work was supported by the CICYT project TIN2007-68091-C02-02

References

[1] M. Greenberg, J. Byington, and D. Harper. Mobile agents and security. *IEEE Communications Magazine*, 36(7):76–85, 1998.
[2] S. Ilarri, J. Serrano, E. Mena, and R. Trillo. 3d monitoring of distributed multiagent systems. In *Third International Conference on Web Information Systems and Technologies (WEBIST'07)*, pages 439–442. INSTICC Press, 2007.
[3] S. Ilarri, R. Trillo, and E. Mena. SPRINGS: A scalable platform for highly mobile agents in distributed computing environments. In *4th Intl. WoWMoM 2006 Workshop on Mobile Distributed Computing (MDC'06)*, pages 633–637. IEEE, 2006.
[4] M. T. Kone, A. Shimazu, and T. Nakajima. The state of the art in agent communication languages. *Knowledge and Information Systems*, 2(3):259–284, 2000.
[5] Y. Labrou. Standardizing agent communication. *Lecture Notes in Computer Science*, pages 74–97, 2001.
[6] S. Macho, M. Torrens, and B. Faltings. A multi-agent recommender system for planning meetings. In *Fourth International Conference on Autonomous Agents, Workshop on Agent-based Recommender Systems (WARS2000)*, 2000.
[7] P. Marques, R. Fonseca, P. Simões, L. Silva, and J. Silva. A Component-Based Approach for Integrating Mobile Agents Into the Existing Web Infrastructure. In *Proceedings of the 2002 Symposium on Applications and the Internet (SAINT'02)*, pages 100–108. IEEE, 2002.
[8] R. T. Meier, J. Dunkel, Y. Kakuda, and T. Ohta. Mobile agents for service discovery in ad hoc networks. In *22nd Intl. Conf. on Advanced Information Networking and Applications (AINA'08)*, pages 114–121. IEEE, 2008.
[9] D. Milojicic, F. Douglis, and R. Wheeler. *Mobility: processes, computers, and agents*. ACM, 1999.
[10] L. Moreau. A fault-tolerant directory service for mobile agents based on forwarding pointers. In *Symp. on Applied Computing (SAC'02)*, pages 93–100. ACM, 2002.
[11] R. Morrow. *Bluetooth: Operation and Use*. McGraw-Hill Professional, 2002.
[12] N. Nehra, R. B. Patel, and V. K. Bhat. MASD: Mobile agent based service discovery in ad hoc networks. In *14th Intl. Conf. on High Performance Computing (HiPC'07)*, volume 4873 of *Lecture Notes in Computer Science*, pages 612–624. Springer, 2007.
[13] F. Ohrtman and K. Roeder. *Wi-Fi Handbook : Building 802.11b Wireless Networks*. McGraw-Hill Professional, 2003.
[14] D. Preuveneers and Y. Berbers. Suitability of existing service discovery protocols for mobile users in an ambient intelligence environent. In *Intl. Conf. on Pervasive Computing and Communications*, pages 760–764. CSREA Press, 2004.
[15] V. Roth and J. Peters. A scalable and secure global tracking service for mobile agents. *Lecture Notes in Computer Science*, pages 169–181, 2001.
[16] C. Spyrou, G. Samaras, E. Pitoura, and P. Evripidou. Mobile agents for wireless computing: the convergence of wireless computational models with mobile-agent technologies. *Mobile Networks and Applications*, 9(5):517–528, 2004.
[17] K. P. Sycara, S. Widoff, M. Klusch, and J. Lu. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous Agents and Multi-Agent Systems*, 5(2):173–203, 2002.
[18] R. Trillo, S. Ilarri, and E. Mena. Comparison and performance evaluation of mobile agent platforms. In *3rd Intl. Conf. on Autonomic and Autonomous Systems (ICAS'07)*, page 41. IEEE, 2007.
[19] O. Urra, S. Ilarri, T. Delot, and E. Mena. Using hitchhiker mobile agents for environment monitoring. In *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS'09)*, pages 557–566. Springer, 2009.
[20] O. Urra, S. Ilarri, and E. Mena. Testing mobile agent platforms over the air. In *1st ICDE Workshop on Data and Services Management in Mobile Environments (DS2ME'08)*, pages 152–159. IEEE, 2008.

[21] O. Urra, S. Ilarri, and E. Mena. Agents Jumping in the Air: Dream or Reality? In *10th International Work-Conference on Artificial Neural Networks (IWANN'09), Special Session on Practical Applications of Agents and Multi-Agent Systems*, pages 627–634. Springer, 2009.

[22] G. Vigna, editor. *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*. Springer, 1999.

[23] J. Zhang, Y. Wang, and V. Varadharajan. A new security scheme for integration of mobile agents and web services. In *2nd Intl. Conf. on Internet and Web Applications and Services (ICIW'07)*, page 43. IEEE, 2007.