A Flexible Mobile-Agent Framework for Accessing Information Systems in Disconnected Computing Environments

Paulo Jorge Marques, Luís Moura Silva, João Gabriel Silva
CISUC, University of Coimbra, Portugal

{pmarques, luis, jgabriel}@dei.uc.pt

Abstract

Over the last few years, the importance of mobile computing as been steadily increasing. While it is important to provide support for accessing databases in disconnected computing environments, many of the information systems being deployed today are using a three-tier architecture. It is becoming vital to find ways of providing reliable access to the business-logic present in the middle-tier, for mobile applications.

In this paper, we present a component-based framework that enables client applications to send mobile agents to interact with application servers in disconnect computing environments. This framework allows the applications to benefit from the advantages of using mobile agents in disconnected-computing environments, and at the same time provides integration between mobile clients and corporate information systems.

1 Introduction

Over the last few years, mobile phones, laptops and personal digital assistants (PDAs) have become commonplace. Laptops and mobile phones are already reshaping the way people work. Companies give workers laptops in order for them to be able to work anywhere, anytime. As mobile devices become more powerful, users are starting to expect to have access to information in any place they are, by using such devices.

At the same time, the information most users access during their everyday work is typically found on corporate databases. The access to these databases is typically done using a client-server architecture, which does not operate well in disconnected computing environments. The problems include:

- Connections being lost and reconnected without any notice, which makes RPC calls to fail.
- Reduced network bandwidth.
- Transactions that require multiple interactions between the client and the server fail due to timeouts or make the applications look sluggish.
- Large costs associated to having a connection up while data is processed on the server side.

To complicate maters, today's corporate information systems (CIS) are typically deployed using a three-tier architecture [1]. Thus, accessing the databases is no longer enough. It is becoming increasingly important to have mechanisms that allow mobile end-devices to access and interact with the business logic present in the middle-tier.

Mobile agents (MAs) are a very interesting approach to software development in disconnected computing environments [2, 3]. A mobile agent consists in a thread of execution that can migrate across the network, performing tasks locally at each host. The advantages of using mobile agents in mobile computing include:

- Connections must only be up for receiving and sending the agents.
- Data must not be transferred from the server to the client: the agents just process it at the server.
- Multiple interactions occur locally at the server, between the agents and server processes.

Nevertheless, currently there is no easy way to deploy mobile agents into the existing three-tier corporate information systems. Current MA platforms center their functionalities on the agents, lacking proper support for applications that just want to make use of agents for particular proposes. Because most of these platforms are generic, they do not adapt well into this application area.

On the client side, installing a complete agent platform for sending and receiving agents is a quite heavyweight approach, preventing the use of a wide range of mobile devices like PDAs and palmtops. The requirements on the client side are normally much more slim. Many of the functionalities needed on a generic multi-hop mobile agent environment are not required for this application domain, where the agents just need to migrate near the application servers and then migrate back.

At the server side, the same considerations apply although the small footprint is not so important. Many of the features found on industrial MA platforms are simply not required. At the same time, many of the available systems lack the necessary mechanisms to support disconnected computing.

In this paper, we present a component-based approach for supporting the mobile-agent concept when developing mobile applications. In this approach, there is no agent platform. Client-side applications become capable of sending, receiving and interacting with mobile agents by using well-defined JavaBeans [4] software components. These components are included in the application, allowing it to become agent-enabled.

On the information system side, a small application receives and manages the incoming agents, allowing them to interact with the application server. This application is built using the components necessary to support the requirements of those agents.

The rest of this paper is organized as follows. Section 2 details the component-based approach for the client and the server side. Section 3 presents related work to the approach and Section 4 presents the conclusions and future work to be done.

2 Component-Based Approach

In our architecture, the applications that are present in the mobile devices become agent-enabled by using simple mobility components. If a Visual Development Environment is used, this can be accomplished by the simple drag-and-drop of components from a component palette and by visually configuring their properties and interconnections. The most important component of our framework is the *Mobility Component*, which allows agents to be sent and received.

On the server side, an application called *Information Agent Server* (IAS) receives and manages the agents allowing them to interact with the application server, which contains deployed Enterprise JavaBeans [5]. The approach is depicted in Figure 1.

In this section, we will detail the design and implementation of the framework. We will concentrate on the client-side implementation, in particular in the Mobility Component and the support of higher-level services.

2.1 The Client Side Design Considerations

While designing and implementing the mobility components at the client side, several considerations where taken into account.

One important issue was to make the framework very light, having a small footprint. This is important since the component middleware is included in the applications, and if several applications are running side-by-side, it is critical that the several instances of the components do not put a large burden on the system. The core Mobility Component conforms to this requirement, being only 54Kbytes in size.

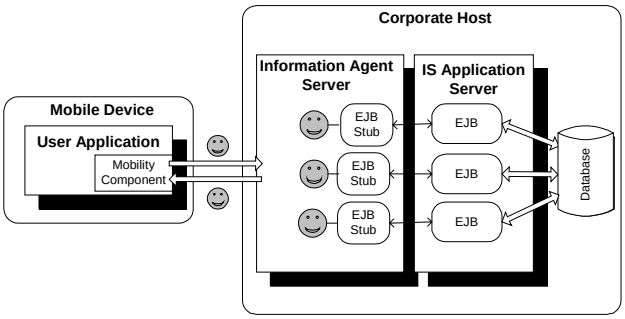

Figure 1 – Agents migrate from the mobile applications into the IS server.

Because the agents being sent and received are only two-hop agents, corresponding to the migration to IAS and the comeback, many of the mechanisms present in general-purpose agent platforms are not required. At the same time, it was important to make framework easily extensible. We support the existence of *services* that can be easily added or removed at design and run-time. Each service is implemented as a JavaBean component. Each component implements a certain functionality that can be made available to agents. Examples include security, persistence and agent/application communication mechanisms. In this approach, each application only makes use of the functionalities it actually requires.

At the client-side, security is much more simpler to accomplish than by using a generic MA platform. When the agent migrates back to the mobile device, its code is already there. Thus, only the state of the agent needs to be received. The application does not have to receive any code from the outside. In addition, because the agents are application-specific, their interface and behavior is well known. All the agents are contained in an "agent-sandbox" inside the mobility component, and have very limited capabilities. Thus, the main security concerns are proper authentication and authorization.

2.2 The Client Side Implementation

In the implementation of the architecture, we decided to make the Mobility Component a little more generic than what was required for the client-side implementation. In particular, the Mobility Component has the ability to generally send and receive mobile agents. By providing such functionality, a wider range of application domains becomes available where to reuse the component. This

includes the implementation of the Information Agent Server.

The Mobility Component is coupled with the application and with the mobility supporting services through events (Figure 2).

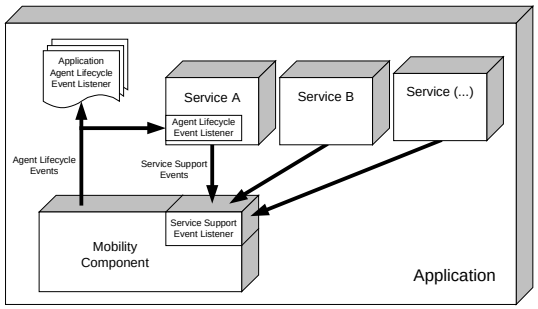

Figure 2 – Interactions between the Mobility Component and the Application.

Two major sets of events exist: *Agent Lifecycle Events* and *Service Support Events*. Agent Lifecycle Events represent changes in the state of a running agent (e.g. when an agent arrives or departs from the application). Service Support Events enable the Mobility Component to incorporate new services at runtime and to be notified if they are no longer available.

Whenever an agent changes its execution state (e.g. arrives, departures or dies), an Agent Lifecycle Event is fired. The application and the mobility services can register their listeners with the Mobility Component, to be notified when one of those events occurs. When an event takes place, the listeners are able to examine the agent responsible for the event and if necessary, call methods on the agent or in other modules of the application. This is especially important for the higher-level services since they can make use of those events to accomplish their functions. For instance, a disk-persistence service can checkpoint an agent to disk after receiving an `OnAgentArrivalEvent` (one of the Agent Lifecycle Events) and remove it after receiving an `AfterAgentMigrationEvent`. For certain events like an agent arrival or an agent migration, each event listener has also the capability of vetoing the event. For instance, a security service may decide not to allow an incoming agent to migrate into the application by vetoing the `OnAgentArrivalEvent`.

Service Support Events exist to allow a service to be incorporated or removed from the system at runtime, and to allow services to be configurable in a Visual Development Environment. A service registers itself with the Mobility Component as a source for those events and, whenever it becomes available, it fires an event that notifies the Mobility Component of the occurrence. Also, if the service needs to be removed from the list of available services, it fires an event requesting to be removed of the available services list. The existence of this event set is extremely import because if allows development tools like BDK [6] to setup adapters between components.

All the features of the Mobility Component are accessible through properties [4]. Thus, for instance, it is possible to configure incoming and outgoing TCP/IP ports for the component, maximum and minimum cache sizes, if they are used or not, maximum number of running agents, and many other characteristics of the component by simply changing these properties in a Visual Development Environment. Figure 4 shows the Mobility Component being used for constructing a simple application.

2.3 Server Side Design Considerations

One important aspect of the component-based approach to mobile agent systems development is the reuse of the basic Mobility Component and the components that implement services. Most of the functionalities needed in the IAS application are in fact agent-related, which are available as reusable components.

Another important issue is that IAS must be safe to use. This is accomplished through an agent-sandbox in the mobility component and through proper authentication and authorization mechanisms. Because in this case there is no need for inter-agent communication mechanisms and because the agents do not perform general-purpose tasks, most of the security problems associated to generic MAs do not apply [7]. The agents contained in the sandbox are only capable of interacting with the application server, and even there, they have to conform to its security policies.

2.4 Server Side Implementation

Figure 3 depicts the architecture of the Information Agent Server application.

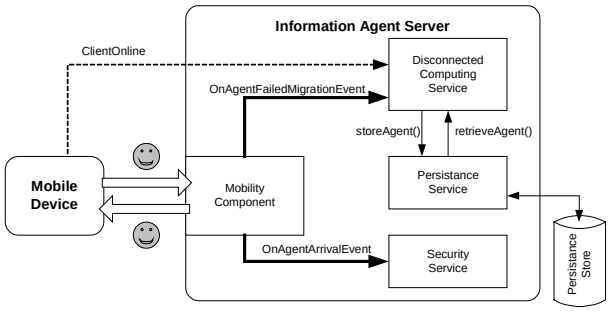

Figure 3 – Information Agent Server.

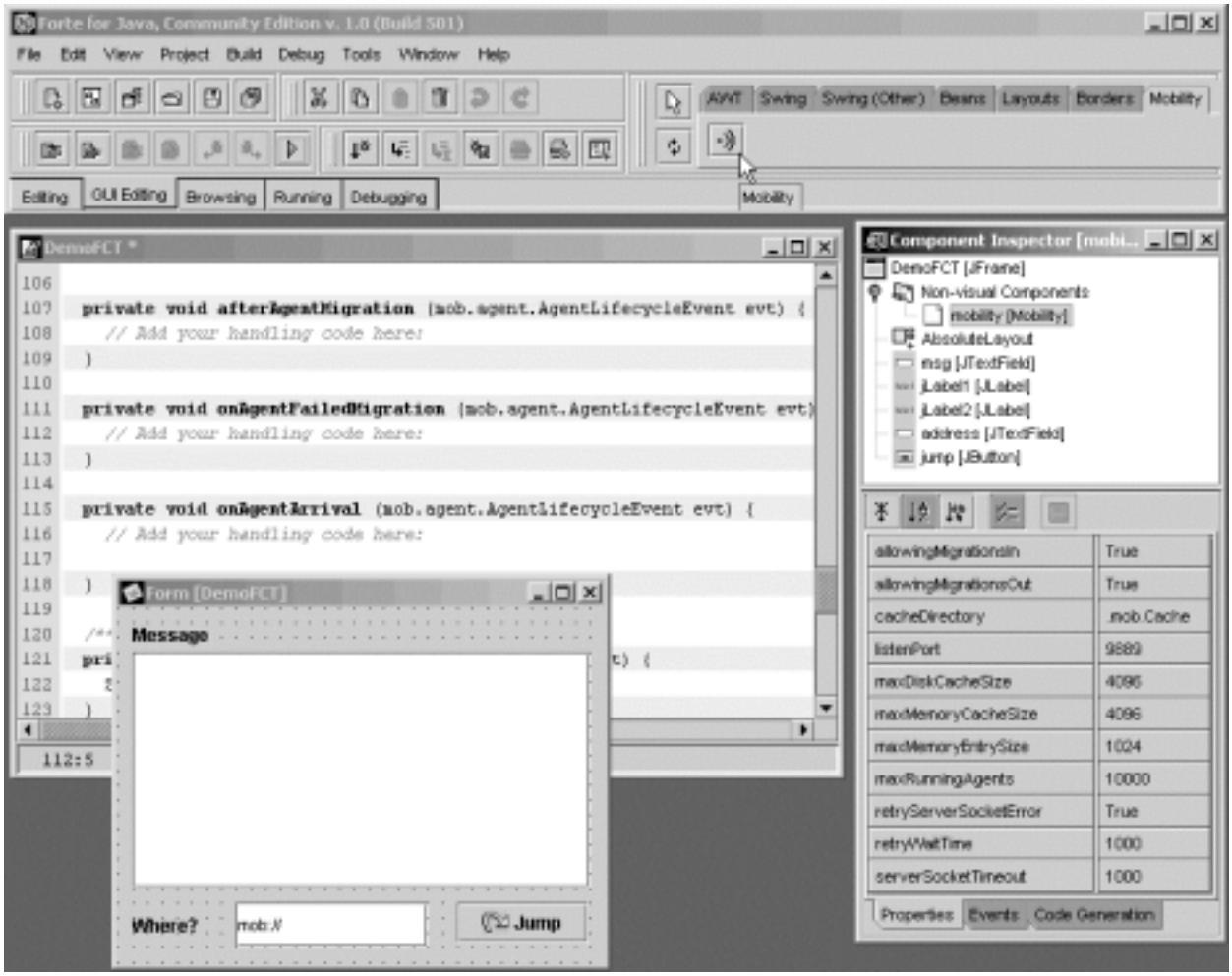

Figure 4 – The Mobility Component being used for developing a demo application.

The Mobility Component is responsible for receiving, sending and managing the running agents. This component is connected with three other components through Service Support Events and Agent Lifecycle Events.

Agents arrive at the application using the Mobility Component. These agents are authenticated though the *Security Service* component, which may prevent an agent from arriving and executing by vetoing its `OnAgentArrivalEvent`.

After arriving, the agents execute, interacting with the application server. The interaction is done using standard EJB home stubs [5] that represent the entity beans and session beans present at the application server.

When the agents are done, they try to migrate back to the client. If an agent fails to migrate because the client is not available, the *Disconnected Computing Service* is notified through an `OnAgentFailedMigrationEvent` (one of the Agent Lifecycle Events). In this case, the agent is stored in disk using the *Persistence Store Service* component. The Persistence Store Service is responsible for keeping in disk the serialized version of the agents, along with their code.

After a failed migration, the Disconnected Computing Service waits for the client to become available. When the client gets online, it notifies the service through a `ClientOnLine` message. When this happens, the agent is retrieved from persistent storage and a new migration is attempted. Pooling is not used since current mobile devices typically use dynamic IPs.

Like in the Mobility Component, all the features of the discussed components are available through properties, and the components can be reused in other application domains.

At the present, we are completing and enhancing the implementation of the components that are used in the Information System Agent Server.

3 Related Work

Oracle and Sybase provide several products [8, 9] for allowing mobile applications to have database access in mobile computing devices. In our approach, we are targeting the business-logic present in application servers on three-tier information systems, and not the access to the databases.

Agent TCL[1] [3] represents early work on supporting mobile agents in disconnected computing environments. Concordia [10], Mole [11] and Magenta [2] are generic mobile agent toolkits that support disconnected computing. While in all this cases, the focus was on building a generic multi-hop agent infrastructure that also supports disconnected computing, in our approach we are concentrating on developing a lightweight and secure supporting infrastructure for accessing corporate information systems in disconnected computing environment.

The OnTheMove project [12] main objective is to create an Application Programmer Interface for use in multimedia mobile applications. The OnTheMove framework combines messages queues and mobiles agents and provides an extensive framework that includes directory services, events, security, replica management, quality of service, and other features.

4 Conclusion and Future Work

In this paper, we have presented a framework that enables mobile applications to send agents to interact with application servers on their behalf.

The applications on the client side become agent-enabled by using a small footprint component that is capable of sending and receiving agents. These components were designed so that they could be reused in other application domains. Due to the reusability objective, another important concern was to make the framework extensible by supporting high-level services, also implemented as components.

On the server side, the agents arrive at an Information Agent Server application, where they have very limited capabilities. They run inside an agent-sandbox and can only interact with the application server. For interacting with the business-logic present in the application server, they must also conform to the security policies of the deployed business components.

We are currently working on the implementation of several services needed to complete Information Agent Server. Afterwards, we plan to conduct a quantitative and qualitative assessment of the advantages and drawbacks of our approach.

[1] Agent TCL is now called D'Agents.

Our longer-term objective in to built a general reusable component framework that enables applications to use mobile agents in a flexible and easy way.

Acknowledgments

This investigation was partially supported by the Portuguese Research Agency FCT, through the program PRAXIS XXI (scholarship number DB/18353/98) and through CISUC (R&D Unit 326/97).

References

[1] R. Orfalli, D. Harkey, and J. Edwards, *Client/Server Survival Guide*, 3rd ed: John Wiley & Sons, Inc, 1999.

[2] A. Sahai and C. Morin, "Mobile Agents for Enabling Mobile User Aware Applications," presented at Autonomous Agents 98, Minneapolis, USA, 1998.

[3] D. Kotz, R. Gray, S. Nog, D. Rus, S. Chawla, and G. Cybenko, "AGENT TCL: Targeting the Needs of Mobile Computers," in *IEEE Internet Computing*, vol. 1, issue 4, pp. 58-67, 1997.

[4] "JavaBeans Specification 1.01," Sun Microsystems, 1997.

[5] R. Monson-Haefel, *Enterprise JavaBeans:* O'Reilly & Associates, 1999.

[6] "The Bean Development Kit," Sun Microsystems, http://java.sun.com/beans/software/bdk_download.html, 1998.

[7] M. Greenberg, J. Byington, and D. Harper, "Mobile Agents and Security," in *IEEE Communications Magazine*, vol. 36, issue 7, pp. 76-85, 1998.

[8] "Oracle8i Lite - Mobile Support," http://www.oracle.com/mobile/o8ilite/index.html?/mobile/o8ilite/content.html.

[9] "Sybase SQL Anywhere Studio," http://www.sybase.com/products/anywhere/.

[10] T. Wash, N. Paciorek, and D. Wong, "Security and Reliability in Concordia," presented at 31st Annual Hawai International Conference on System Sciences, Hawai, 1998.

[11] M. Strasser, J. Baumann, and F. Hohl, "Mole - a Java based mobile agent system," presented at 2nd ECOOP Workshop on Mobile Object Systems, Austria, 1996.

[12] E. Kovacs, K. Rohrle, and M. Reich, "Mobile Agents OnTheMove - Integrating an Agent System into the Mobile Middleware," presented at ACTS Mobile Summit 1998, Rhodos, Greece, 1998.