Integrating Mobile Agents into Off-the-Shelf Web Servers: The M&M Approach

Paulo Marques, Raul Fonseca, Paulo Simões, Luís Silva, João Silva
CISUC, University of Coimbra, Portugal

pmarques@dei.uc.pt

Abstract

The mobile agent paradigm provides a new approach for developing distributed systems. During the last two years, we have been working on a project that tries to overcome some of the limitations found in terms of programmability and usability of the mobile agent paradigm in real applications. In the M&M framework there are no agent platforms. Instead applications become agent-enabled by using simple JavaBeans components. In our approach the agents arrive and departure directly from the applications, interacting with them from the inside.

In this paper we discuss our experiences on integrating these components into off-the-shelf web servers, enabling them to receive and send agents. Our approach involves wrapping the components inside a Java servlet that can be included in any web server supporting the Servlet Specification. This servlet enables the servers to receive and send agents that can query local information, and also enables the agents to behave as servlets themselves. This approach departs from the current available systems because it enables any existing web server that supports the servlet specification to send and receive agents in a straightforward way.

1 Introduction

Mobile agents are small threads of execution that are able to migrate from machine to machine, performing operations locally [1]. One very interesting application area for mobile agents is internet computing. Mobile agents provide a very attractive paradigm for this area. The agents can be launched from a machine, navigate from web-site to web-site, collecting information or performing transactions, finally returning home with the goods or results. This scenario is especially attractive when we consider the proliferation of wireless mobile devices that is currently taking place. It is easy to imagine a scenario where a user can launch a news gathering agent into the web, shutdown the device and reconnect hours later, collecting the results. Or even different e-commerce scenarios where the agents find the best deals for their users in an autonomous way.

Currently available systems that provide integration between mobile agents and web servers follow basically two models. The first one involves installing an agent platform that is unaware of the web-server, or has little integration with it [2,3,4]. The agents migrate to and from the agent platform and interact with the web server as if they were just normal clients, with the difference that they are local. Although this approach is appropriate for operations like querying information on the host, or monitoring when certain changes happen, it quite limits the functionality that can be implemented on the agents. For instance, it is quite hard for the agents to publish information on the site, or to extend the functionality of the servers by migrating agents into them, or even having the agents represented in a web page of the server for their users to remotely interact with them.

The second approach consists in developing a costume-made web server that is also able to host agents [5,6]. The problem is that typically the web sites are already up and running, and do not want to replace their existing infrastructure. Also, in most cases these agent-enhanced web servers do not have the robustness or scalability needed for production-running sites. So, it would be quite difficult for a web site manager to accept replacing its industrial-strength web infrastructure for a technology that follows one of the above approaches.

Thus, although mobile agents provide an attractive conceptual framework for internet-based computing – small threads migrating from server to server, performing their functions, there are still many difficulties that must be addressed. These difficulties are currently preventing the widespread adoption of the technology. Some of the key problems include: security, user and provider psychological resistance on using agents, infrastructure integration, interoperability and reliability. This paper deals primarily with the infrastructure integration problem.

During the last two years, we have been working on a project that tries to overcome some of the limitations found in terms of programmability and usability of the mobile agent paradigm [7]. In the M&M framework there are no agent platforms. Instead applications become agent-enabled by using simple JavaBeans components. The applications can be developed using current object-oriented approaches and become able of sending and receiving agents by the drag-and-drop of binary software components. In our approach the agents arrive and departure directly from the applications, interacting with them from the inside.

In this paper we discuss our experiences on integrating the framework components into off-the-shelf web-servers, enabling them to receive and send agents. Our approach involves wrapping the components inside a Java servlet that can be included in any web server supporting the Servlet Specification [8]. This servlet enables the servers to receive and send agents that can query local information, and also enables the agents to behave as servlets themselves. We currently have experimented the framework with several web servers, inclusively having the security mechanisms of the framework correctly running and integrated with the security architecture of the server.

One of the most important characteristics of our approach is that it does not force the deployment a stand-alone agent server that is not integrated with the web server, nor does it require a specialized costume-made web server. We provide a framework that is able of using existing web infrastructures, giving them the capability of using agents in their operation.

The rest of the paper is organized as follows. Section 2 gives an overview on the M&M framework. Section 3 discusses the developed architecture, and Section 4 presents the conclusions of the work.

2 M&M Overview

The most distinctive characteristic of the M&M framework is that there are no agent platforms. Instead, agents arrive and leave from the applications they are part of, not from agent platforms. The applications become agent-enabled by incorporating well-defined binary software components [9] into their code. These components give the applications the capability of sending, receiving and interacting with mobile agents. The applications themselves are developed using the current industry best-practice software methods and become agent-enabled by integrating the mobility components. We call this approach *ACMAS – Application Centric Mobile Agent Systems* — since the applications are central and mobile agents are just a part of the system playing specific roles.

The consequences of ACMAS are as follows:

- It is not necessary to design the whole application around agents. Agents are sent back to middleware, in pair with other distributed programming technologies.
- Security is integrated with the application security framework, rather than being completely generic.
- Agents interact directly with the applications from the inside. This eliminates the need to setup interface agents and configure and manage their security policies.
- There is no agent platform to install and maintain. Although there are still distributed applications to install and manage, this is much simpler than managing a separate infrastructure shared by a large number of distributed applications with different policies and requirements.
- The end-user sees applications, not agents. In this way, the acceptance of applications that use mobile agents is increased since what the end user sees is the added value functionality, not the agents.

The M&M framework was implemented using the JavaBeans component framework, and is centered on the so-called *Mobility Component*. This component provides the basic support for agent migration and management, and an extensibility mechanism that allows other components to connect to it [10]. These other components may implement functionalities like different inter-agent communication mechanisms, security, persistence and others.

Our main concern was to make the core component small and efficient, and at the same time provide a powerful extensibility mechanism that allows higher functionality to be connected to it. In fact, the Mobility Component is only 56k in size, and has a very lightweight runtime footprint. The extensibility mechanism allows us to have a rich reusable component palette, which provides different components that are selectively used on each application being developed. If there is no component that implements the functionality needed for a certain application, it is always possible to code it for the application, and reuse it when creating new applications.

The complete discussion of the M&M project and its associated framework is beyond the scope of this paper but the interested reader can refer to [7,10,11] for additional details.

3 Agent Enabling Web Servers

Our interest in building support for mobile agents in web servers arouse from the necessity of validating how easy (or not) was to agent-enable existing applications by using the M&M framework. Web servers, and in particular the creation of web agents, appeared to be an interesting application field because the paradigm seams so fit for using on the web.

In our view, the main requirements for integrating mobile agents into web servers are:

- It should be possible for the agents to behave as a web resource (i.e. publish information). A user should be able to use a web browser to access and interact with the agents, that would be dynamically generating the web pages.
- The agents should be able to query local information present on the web server.
- It should be possible to use the existing web-infrastructure (i.e. the fact of using mobile agents should not force the existing servers to be changed).
- If possible, certain agents (administration agents) should be able to perform management operations on the server. This last requirement was based on our interest in studding the usefulness of mobile agents for distributed network and application management.

The key idea for building server-independent support for mobile agents was to wrap the necessary components, namely the Mobility Component and the Security Component, inside of a servlet [8].

The servlet technology provides a simple mechanism for extending the functionality of a web server, allowing URIs to be associated with object instances. These instances are called servlets, and are able to process HTTP requests sent to them. Currently there are many web-servers supporting the Servlet Specification [8], and there are many stand-alone servlet engines that can be connected to the web servers for providing servlet functionality [12, 13, 14].

3.1 Architecture

To meet the previously discussed requirements, we came up with the architecture depicted in Figure 1. The servlet that encapsulates the framework components is configured and included as a resource on the web server. What this means is that whenever an HTTP request is made for a certain URI, that request is forward to the servlet (*Mobility Servlet Container*).

The Mobility Servlet Container, or simply *container*, performs several functions. Because the M&M extensibility mechanism is based on events, which notify the interested components of any changes in the running state of the agents, it is possible for the container to know the state of every agent in the system, and to publish that information on the web. This means that a user is able to remotely interact and control the agents presently running on the site.

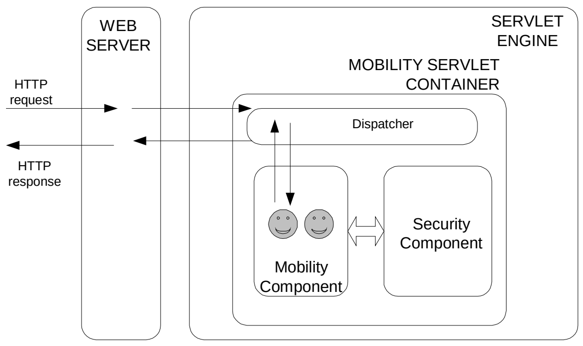

Figure 1 – The Mobility Servlet Container

The information published by the container on the web is accompanied by a link, which includes the identity of each agent. Whenever a user clicks on such a link, the GET request is forwarded to the container, which in term forwards it to the appropriate agent. The container is able to do this because when an agent arrives, it receives the corresponding agent and saves a reference to it. Also, when the agent migrates or dies, the corresponding events are fired, enabling the container to garbage-collect the references.

One important point is that the web server may be decoupled from the servlet engine, and from the servlet itself. In this case, the function of the web server is simply to provide a mapping between URIs and the resources, forwarding the requests to the appropriate servlets. Each request that corresponds to an interaction with an agent is forward to the container, which then passes it to the appropriate agent. It should be noted that it is not necessary to decouple the web server from the servlet engine. If the web server supports the Servlet Specification by itself, then the container may be installed and configured on the web server itself.

We will now discuss how the agents see their interaction with the web server. The M&M framework provides the concept of *services for agents* [10]. What this means is that an agent on arriving at a host can query which are the currently available services, and request an object implementing that service interface. That idea was used in our implementation. When an agent arrives at a web server, it may not only query the local web server, but it can also ask for a service instance that allows it to behave as a servlet, and publish information.

When an agent requires an object that allows it to publish information, the object that is passed actually requires the agent to implement the servlet interface. Thus, any HTTP request made to an agent contains the full information about the request. This includes not only the IP of the client, MIME-types accepted but also session information. This is especially important since allows the agents to distinguish between different clients, and act accordingly. Also, when the agents request the service instance, they can require or deny that they are listed online. The main point is that the agents are able to seamlessly register their interest in processing certain HTTP requests, and examine all the characteristics of those requests.

Finally, one other interesting point of this approach is that because in the M&M framework the agents arrive and interact with the applications from the inside, the agents have access to the internal objects of the applications. In our case, what this means is that it is possible for the agents to access management information present inside of the web server. This also enables the agents to perform maintenance tasks on it from the inside. This feature is especially important for us because one of the aims of our project is to study the applicability of the framework for developing distributed management applications. Although the way the agents access the information varies from web server to web server, the potential of having such functionality is quite important.

3.2 Security

In our container, the Security Component is instantiated and provides several protection services for the agents and the host.

This component allows the agents to migrate between hosts using SSL, which prevents tampering and eyes dropping on the contents of the agents. It also implements a fine-grained authorization mechanism that guaranties that only the agents with the correct permissions can perform certain operations, like reading directly from disk, or connect to other hosts in the network. Finally, the component implements cryptographic primitives that allow secure protocols for information gathering and comparison-shopping [15], that use mobile agents, to be implemented in an easy way.

The main difficulty with security is that only one security manager can be instantiated at one time in a JVM. Because the framework components are running inside third-party software, which already has a security manager running, the framework must rely on it. Fortunately the Java 2 platform standard [16] redefined how security is implemented and verified in Java. The methods of the security manager no longer have to be overridden. In the Java 2 architecture there is a delegation mechanism that allows the runtime permissions of the classes to be checked in an autonomous way. Thus, as long as the servlet engines do not change the security managers in a way that they do not conform to the Java 2 model, our security component is able to check the runtime permission of the agents, and their actions. We found that this is the situation in most of the currently available engines.

In our approach, the main problem to be solved concerning security is resource control. Since the Java platform does not include any methods for this, it is a complicated issue. We are currently investigating the possibility of using third-party resource control libraries with our system, and its implications in terms of runtime penalty.

3.3 Current Perspective

We currently have experimented with the framework in several web servers and servlet engines, with very positive results. We have tested the system with W3C Jigsaw web server [17], Apache's Tomcat [13], Allaire's JRun [14] and Sun's JWS [18]. When a servlet engine was used, it was tested using the Apache web server [12] as front-end. Our main conclusion is that it is quite straightforward to enable an off-the-shelf web server to receive agents, as long as it supports servlets. This is the common case on the currently available servers.

Our experience is that the M&M framework provides a good approach for integrating mobile agents into existing web infrastructures. The main limitation found with the framework has to do with resource control. Currently the framework is only usable in a secure way, on an intranet or on an extranet. On these types of networks it is possible to create accounts and use the authentication, authorization and logging mechanisms present in M&M for holding the users accountable. We believe that in the future standard resource control mechanisms will be introduced in Java, easing the problem.

4 Conclusion

In this paper we have presented our experiences on using the M&M framework for developing web-based mobile agents. The M&M component framework can be added into existing applications for agent-enabling them, providing the support needed for receiving and sending agents in an easy way. In this work, we have built an architecture that allows any web server that supports the servlet specification to receive agents. The main features of the architecture are:

- Any web server that supports the Servlet Specification is able to receive and send agents. This allows the deployment of mobile agent technology in existing web infrastructures.
- The execution of the agents is restricted by proper authentication and fine-grain authorization mechanisms, so long as the existing security manager has not been modified in a way that is not

compatible with the Java 2 security delegation mechanism.

- The agents are able of processing HTTP requests, having session information, as well as acting as regular servlets.
- It is possible to dynamically load new services, adding new features at run time. This makes the approach very configurable and capable of addressing different requirements of different sites.
- It has a small footprint and a lightweight execution environment.

We believe that our solution constitutes a good approach for agent-enabling existing infrastructures. There is still a long way to go in order to address all the problems mentioned in the first section, but at the present time, a solution as the one presented here is quite appropriate for being used on an intranet or extranet, where the users can be held accountable.

Acknowledgments

This investigation was partially supported by the Portuguese Research Agency – FCT, through the program PRAXIS XXI (scholarship number DB/18353/98), the M&M project (project reference POSI/33596/CHS1999), and by CISUC (R&D Unit 326/97).

References

[1] J. White, “Telescript Technology: Mobile Agents”, General Magic Whitepaper, in *Software Agents*, AAAI/MIT Press, 1996.

[2] C. Dharap, M. Freeman, “Information Agents for Automated Browsing,” in *Proc. of the ACM CIKM’96*, Rockville, USA, 1996.

[3] W. Theilmann, K. Rothermel, “Domain Experts for Information Retrieval in the World Wide Web”, in *Proc. 2nd Int. Workshop on Cooperative Informative Agents (CIA'98)*, 1998.

[4] V. Roth, M. Jalali, R. Hartmann and C. Roland, “An Application of Mobile Agents as Personal Assistents in Electronic Commerce”, in *Proc. 5th Conference on the Practical Application of Intelligent Agents and Multi-Agents (PAAM’2000)*, Manchester, UK, April 2000.

[5] S. Fünfrocken, “How to Integrate Mobile Agents into Web Servers”, in *Proc. IEEE ICE’97 Workshop on Collaborative Agents in Distributed Web Applications*, Boston USA, June 1997.

[6] G. Neumann, “High-level Design and Architecture of an HTTP-Based Infrastructure for Web Applications,” in the *World Wide Web Journal*, vol. 3(1), 2000.

[7] P. Marques, P. Simões, L. Silva, F. Boavida, J. Gabriel, "Providing Applications with Mobile Agent Technology", in *Proc. 4th IEEE International Conference on Open Architectures and Network Programming (OpenArch'01)*, Anchorage, Alaska, April 2001.

[8] Sun Microsystems Inc., “The Servlet Specification 2.3”, http://www.javasoft.com/servlet.

[9] Sun Microsystems Inc, “JavaBens Specification 1.01”, http://www.javasoft.com/beans.

[10] P. Marques, L. Silva, J. Silva, “Addressing the Question of Platform Extensibility in Mobile Agent Systems”, in *Proc. International ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce (MAMA'2000)*, Wollongong, Australia, December 2000.

[11] P. Marques, L. Silva, J. Silva, “Going Beyond Mobile Agent Platforms: Component-Based Development of Mobile Agent Systems”, in *Proc. 4th International Conference on Software Engineering and Applications (SEA’2000)*, Las Vegas, USA, November 2000.

[12] The Apache Consortium, “The Apache web server”, http://httpd.apache.org/.

[13] The Apache Consortium, “The Jakarta project”, http://jakarta.apache.org/.

[14] Alaire Corporation, “The JRUN server”, http://www.jrun.com/Products/JRun/.

[15] G. Karjoth, N. Asokan, and C. Gülcü, “Protecting the Computation Results of Free-roaming Agents,” in *Proc. Second International Workshop on Mobile Agents (MA' 98)*, Stuttgart, Germany, 1998.

[16] Sun Microsystems Inc., “The Java 2 Platform”, http://www.javasoft.com/j2se/.

[17] W3C Consortium, “The Jigsaw web server”, http://www.w3.org/Jigsaw/.

[18] Sun Microsystems Inc, “The Java Web Server”, http://www.sun.com/software/jwebserver/index.ht ml.