A Component-Based Approach for Integrating Mobile Agents Into the Existing Web Infrastructure

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

pmarques@dei.uc.pt

Abstract

Mobile agents provide a new abstraction for deploying functionality over the existing internet infrastructure. 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 this paper we present an architecture that allows currently available web servers to become capable of sending and receiving agents in an easy way. By using this approach, existing web infrastructure can be maintained, while gaining a whole new potential by being able to make use of agent technology. 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.

We currently have used the framework with several existing commercial web servers, inclusively having the security mechanisms of the framework correctly running and integrated with the security architecture of the server.

Keywords: Mobile agents, Web agents, Components

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. A user can launch an agent into the web, shutdown the device and reconnect hours later, collecting the agent with the results. Some key applications for agents in internet computing include:

- Information gathering agents, which collect information from different web sites or distributed databases, finally presenting it to its owner.
- Shopping agents, which look for the best deals for their owners, perform commercial transactions, and present the best results found, so that their owners can make a decision.
- Management agents, which carry information into selected web sites or databases and make sure that all the distributed web-infrastructure is up-to-date.
- Monitor agents, which migrate into selected web sites and monitor some information (like stock options), warning the owners when certain events happen or even performing some actions on those events.

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, infrastructure integration, interoperability and reliability.

1.1 Key Problems

Security. If one wants to deploy mobile agents into the world-wide-web, security is a critical issue that must be carefully considered [2,3]. There are many points to examine when it comes to mobile agent security. Because the agents are going to arrive at a host that probably knows nothing about them, there must be mechanisms that prevent the agents from damaging the host or access information that they do not have permissions to. Also, because the agents are going to execute in an open environment, on machines that they may not known them very well, they are extremely vulnerable to attacks from those. The hosts can steal information from the agents; make them perform actions they did not voluntarily wanted to; or even misguide them into give false information to other entities. Finally, the agents must be protected from attacks of other agents running in the same host.

Currently, the mechanisms needed for protecting the hosts from the misbehaved agents and the agents from attacks of other agents are well known [2, 4]. These mechanisms are mostly based on proper authentication and authorization. Even so, today there is a major technical problem on protecting the hosts from the mobile agents. Most mobile agent systems available today are implemented in Java [5]. This is due to the fact that Java already provides many mechanisms, like dynamic code loading and object serialization, which allow an easy and efficient implementation of the paradigm. And although the Java 2 platform [6] has a fine-grained security model, currently there is no feasible way of doing resource control. Thus, it is not possible to make sure that an agent will not perform a DoS attack on the server, by allocating a huge amount of memory, by using all the available network bandwidth or even by burning CPU cycles.

Protecting the agents from the hosts is technically very difficult [7]. Because the agents are executing on a host, the host has access to all the state and code of the agent. Although there are some promising approaches for solving this problem, like computation with encrypted functions [8, 9] and code obfuscation [10], the problem is still far from being solved.

Psychological Resistance. The term mobile agent, and also the term mobile code have very strong negative connotations. A user is afraid of installing an agent platform that is able of receiving and executing code without his permission. The first thing a user associates with mobile agents is computer viruses, even though mobile code is currently present in technologies like Java, in particular in RMI and JINI. The main difference is that in those technologies the user is shielded from the existence of mobile code, while with the currently available platforms, the mobile code and the agents are widely visible to the user. This prevents the adoption of the technology, even though it can bring added value. For the interested reader, this subject is more extensively discussed in [11].

It is not only the user that is resistant to install an agent platform on his machine. The web host administrators must also be convinced of the value of letting people send agents into their machines. Besides being potentially dangerous to let people run agents on the machines, spending resources and opening the system to more vulnerabilities, there is also the economical side of the question. From the user point-of-view, it may be interesting to have a comparison-shopping agent that roams from host to host looking for the best deal, but for the hosts providing shopping services, this may not be desirable. This problem already happens with static client/server comparison-shopping search engines, which are constantly getting blocked.

Infrastructure Integration. Another relevant issue is how a web site should integrate a mobile agent platform into its infrastructure. Currently available systems follow basically two approaches.

The first one involves installing an agent platform that is completely unaware of the web server. 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 available approach consists in developing a custom-made web server that is also able to host agents. The problem is that typically the web sites are already up and running, and do not want to replace their existing infrastructure. Also, typically these agent-enhanced web servers do not have the robustness or scalability needed for production-running sites. Thus, it would be quite difficult for a web site to accept replacing its industrial-strength web infrastructure for a technology that follows one of the above approaches.

Interoperability. Another serious problem preventing the adoption of the technology is interoperability. Currently there are over seventy-two known implementations of mobile agent platforms [5], and none is able of receiving agents from another. Standards like MASIF [12] and FIPA [13] only cover the interfaces needed for agent and system management, not how to migrate an agent between different systems. Thus, if a web host is going to support an agent platform, which one should it support? Supporting only one locks the number of potential clients of the service. Supporting more than one means additional costs, problems and vulnerabilities.

An approach like the one followed in [14], which abstracts the common functionalities that exist in most agent platforms into a middleware layer, allowing agents from different systems to migrate into a common system, helps to ease the problem. Nevertheless, this approach does not allow the agents to take advantage of the more advanced features of each platform.

Reliability. Another very important question is reliability. If a user is going to send an agent into the web, many things can go wrong that can make the agent to be lost. A simple server crash may kill all the agents that are running there. Even a routine operation like a server shutdown may lead to the loss of the agents running on the server.

There are currently many mechanisms that can be applied for ensuring that the agents do not get lost, like persistent storage and fault-tolerance techniques [15]. Nevertheless, it is important to carefully consider the reliability requirements when deploying an agent infrastructure on the web.

1.2 Outlook

In our opinion, the mobile agent paradigm provides a very good conceptual model for developing distributed internet applications. Nevertheless, there are some very important problems that must be addressed.

We believe that security, from the point of view of the host, is solvable in the near future, depending mostly on the adoption of basic resource control methods in the Java platform. Protecting the agents from the hosts, it is a complicated problem in the general case, but there are approaches that can be used in web agents [16, 17]. These approaches give some security guaranties for the agents running on the web, by carefully considering the requirements of those agents in the internet domain.

Regarding the resistance of the users on using mobile agents, it is necessary to provide a stronger focus on the applications that use agents, and not on the agents themselves [11,18]. We believe that the user doesn't even need to be aware of the agents or the agent platforms. What he needs to see and interact with are the applications. On the other hand, convincing the web hosts to introduce the technology into their sites is basically a question of market, and maturation of the agent technology. When the technology comes to a point where the providers can be assured that there is no danger in deploying such a framework on their infrastructure, they will give such functionality to their users. This will allow them to differentiate from the competition, providing a better service to the customers.

Infrastructure integration, interoperability and reliability are serious technical problems that must be addressed. This is just part of maturing process of the technology. In this paper we address the infrastructure integration problem.

1.3 Our work

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 [11]. 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 [19]. 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.

Thus, on this paper we are basically addressing the infrastructure integration problem. Our approach does not involve deploying 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.

Our framework also provides a very strong security model, with authentication and authorization mechanisms that control incoming agents, and cryptographic primitives that are useful to protect the integrity and confidentiality of the agents. This is essential if an infrastructure is going to be deployed on an open environment.

Finally, our framework also allows the user distrust on mobile agents to be addressed. With our framework the user does not even has to be aware that he is using mobile agent technology. He only should be aware of the positive results of using this technology.

Concerning this paper, one important point should be made. We specifically tried to tell the complete story on how this research was conducted. Our first approach, the problems and limitations found, how they were solved, and the lessons we learned from that. We believe that this is more informative than simply giving the final architecture without discussing the problems found while trying to make a system work.

The rest of the paper is organized as follows. Section 2 gives an overview on the M&M framework. Section 3 explores our first approach. Section 4 discusses the final architecture. Section 5 presents some performance results. Section 6 analyses the related work. Section 7 concludes the paper.

2 M&M Overview

The most distinctive characteristic in our approach 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 [20] 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/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.
- It is simple to program. The programmer only needs to visually drag-and-drop the necessary components from a component palette and configure their properties and interconnections.

The M&M framework was implemented using the JavaBeans component framework [20], 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 [21]. These other components may implement functionalities like different inter-agent communication mechanisms, security, persistence and others.

One very important aspect of the extensibility mechanism is that it is based on an event model – `AgentLifecycleEvents`. After a high-level service component has registered with the Mobility Component, it is notified whenever some state transition occurs in an agent. For instance, when an agent arrives, there is an `onAgentArrivalEvent`. Every listener is able of examine the agent, interact with it, and can veto the corresponding event. As an example, consider the security component. On being notified that an agent is arriving, it can verify its credentials and examine its state. If it finds the agent not to be trusted, it can veto the event, prohibiting the agent from arriving.

The complete discussion of the M&M project and its framework is beyond the scope of this paper. Interested readeres can refer to [11,21,22] for additional details.

3 Integrating M&M into web servers – A first approach

3.1 Motivation

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.

Our first experiences were made with the Jigsaw web server [23]. We were interested in Jigsaw because it provides an extensible architecture where new services can easily be introduced by implementing a simple adapter. In the case of our framework, this seamed to be perfect since this adapter could be used for housing the Mobility Component.

For this particular system, we had three requirements:

- 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.
- If possible, the 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.

3.2 Architecture

To meet the above requirements we came up with the architecture depicted in Figure 1.

Jigsaw provides a class that must be overridden for each web-resource. This class is called an `HTTPFrame`. For making a web-resource available, its corresponding object must be associated with an URI, which is done by configuring the Jigsaw resources.

The `MobilityWrapper` is a class that overrides the `HTTPFrame` and houses the Mobility Component. This wrapper also listens to the `AgentLifecycleEvents`. Thus, at any given time it knows the state of every agent in the system, and publishes this information into a URI. This means that a user accessing the web site is able of seeing which agents are presently running on the server.

The information published by `MobilityWrapper` about each agent is accompanied by a link, which includes the identity of the agent. Whenever a user clicks on such a link, the agent identity is passed on the GET request to the `MobilityWrapper`. It recognizes that this is a request that is to be handled by an agent, and forwards it to the currently running agent whose identity matches the parameter. The wrapper is able to do this because when an agent arrives, it receives the corresponding event and saves a reference to it. Also, when an agent migrates or dies, the wrapper also receives an event and is able to garbage-collect that reference. The bottom line is that it is possible for a user to interact with the agents currently running on the web server by simply accessing a starting page.

Another 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

Jigsaw. 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. In fact, one of our first prototypes was a simple management application that used mobile agents to collect information from the web servers, and performed simple administration operations on them.

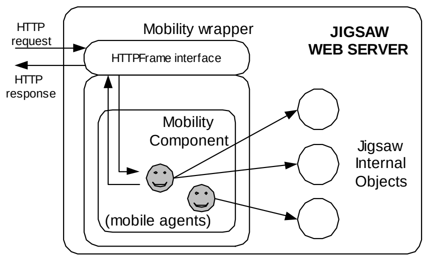

Figure 1 – Using M&M with Jigsaw

3.3 Lessons Learned

Our main conclusion from this first experience was that it is quite straightforward to integrate mobile agents into Jigsaw, by using M&M. Another very important point for us was that having the agents interacting with the applications from the inside, having access to its internal state, opens many possibilities in terms of distributed application management. Nevertheless, considering what was needed to do real-world deployment of agents in web servers, there were still two very significant shortcomings on our approach that needed to be addressed.

The first problem was that we were "agent-enabling Jigsaw". The approach was not general, and was not applicable to other web servers. As it was discussed on the introduction, a content provider will not typically replace its web infrastructure for simply adding a feature. We believed that there should be a more general way of integrating the mobile agents into the web servers.

The second problem concerned security. Although the M&M framework provides components for security, at this point we were making the experiments with security turned off. The main reason for this was that Jigsaw already has a `SecurityManager` instantiated and taking care of its security. Because Java only allows one security manager to be running at one time, we were not sure if we could turn security on, and it would work. The component framework was thought with that in mind, but at that time, it was still an open issue. Nevertheless, security was an important point that needed to be addressed.

Finally, we realized that the way the wrapper was interacting with the agents was not the most appropriate one. In this implementation, the requests were directly forward to one of the methods implemented in the agent. The agents had no saying on if they wanted to process the requests or not, or if they wanted to be visible on the web site.

The above points motivated us to develop an approach that was web server independent, but at the same time allowed the agents to arrive and departure directly from the server, and benefit from all the its associated advantages. Finally, the agents should be able to register their interest on processing HTTP requests, and be able to examine the characteristics of those requests.

4 The Mobility Servlet Container

The key idea to build server-independent support for mobile agents was that the `HTTPFrame` interface was not providing much more than what could be provided by the Servlet Specification [19]. The `HTTPFrame` is only providing a hook for mapping an URI to an object inside of the web server that can respond to the HTTP requests. This can be accomplished with a servlet.

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, and there are many stand-alone servlet engines that can be connected to the web servers for providing servlet functionality [24, 25, 26]. Thus, migrating the wrapper into a servlet container would allow us to run the framework in any web server or servlet engine that supported the specification. Several issues were brought up considering this migration.

Our first concern was if it would not be too heavy to run the framework on a servlet engine. Our expectation was that it would not be, since the main component, which was the one being used, has a very small footprint and while running is also very lightweight, offering good scalability.

The second point that we considered was that currently there is no uniform manner by which the agents can access the information on the web server. Making the agents behave as data sources associated with a URI is easy, since the servlet can forward the requests to the appropriate agent. The problem arises when an agent has to access the information stored locally on the web server. The most straightforward approach, and the one that we adopted, was to have the agents read the information as just an ordinary HTTP client. Although there is a small performance penalty, it is not very significant since the agents and the data source are in the same machine, and the loopback interface provides very large bandwidth.

4.1 Architecture

The implementation of the servlet container follows the same base guidelines of the Mobility wrapper, but with some important changes.

First, 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 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 Mobility Servlet Container, which then passes it to the appropriate agent. Secondly, the Security Component of our framework is instantiated and running, providing security features for the running agents, and for the host. Figure 2 shows the approach. 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.

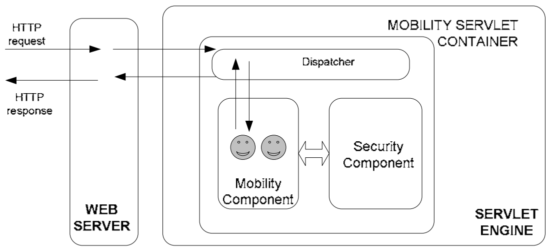

Figure 2 – The Mobility Servlet Container

The biggest change on the architecture is not visible on the forwarding process taking place. Rather, it is on how the agents see and interact with web resources.

The M&M framework provides the concept of *services for agents* [21]. 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 that the agent to implement the servlet interface (Figure 3). 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 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.

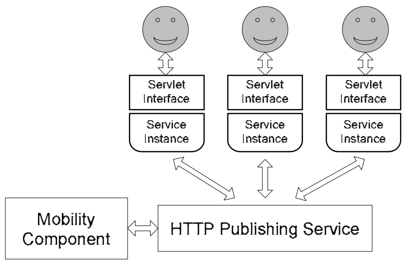

Figure 3 – Agents are able to behave as servlets

4.2 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 eavesdropping 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 [16], that use mobile agents, to be implemented in an easy way.

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.

4.3 Adding new services

On interesting aspect of our system is that it allows different services to be instantiated and made available for the agents at runtime. Its makes the framework very flexible for implementing different features on different web sites.

For instance, let's suppose that a programmer wants to implement a marketplace for agents, where the agents negotiate between themselves, and consult and publish information on the web site. The programmer can use the basic architecture described here and configure the Mobility Component to load one or more of the different available components that implement several inter-agent communication mechanisms. This allows the agents to negotiate while executing on the web site.

4.4 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 [23], Apache's Tomcat [25], Allaire's JRun [26] and Sun's JWS [27]. When a servlet engine was used, it was tested using the Apache web server [24] as front-end. For experimenting with the framework, we have built some prototype applications. Two of the most interesting applications were:

- An electronic commerce portal, based on agents. Basically, the user can login into a portal, being authenticated, and then specify a certain number of items that he wishes to buy. An autonomous agent is created for that user, which navigates though several agent-enabled web sites, collecting information (prices) concerning those items. At any given time the user can signoff, since it does not affect the operation of the agent. When the user is logged on, it can see where the agent currently is, or if the agent has already returned home, he can see what were the items that the agent found. For making transactions, two modes of operation are provided: the user may initially give a certain credit to the agent, which is used to purchase the least expensive items of the list; or the user does not give any credit to the agent, and just sees the offers that the agent found. In this case, when the agent returns home, the user can choose the items he wants the agent to buy, re-dispatching the agent, or make the purchases manually.

- Another application that was developed was a distributed site-indexing system (a mobile agent-based crawler). In this application, the user specifies a site that he whishes to be indexed, and an agent jumps to that site performing the indexation locally. Because building the complete index of a site requires that all the pages of the site to be accessed, using an agent that performs the operations locally is much less expensive in terms of time and bandwidth than bringing the complete site to the local machine. The performance results of this application are discussed in the next section.

Our experience is that the M&M framework provides a nice approach for integrating mobile agents into existing web infrastructures. One of the most fascinating characteristics of the approach is that after having the basic infrastructure deployed (the container servlet), any new functionality can be easily introduced into the existing web infrastructure. All that is required is simply to code the required agents (and/or the agents behaving as servlets), and send them to migrate to the target web servers.

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 resource control mechanisms will be introduced in Java.

5 Performance Results

We will now examine some performance results of the distributed agent-based crawler application. Although these are not extensive tests, they are useful to show the performance gains that can be expected by using a mobile agent-based approach while building distributed applications.

5.1 Experimental Setup

For understanding how well the mobile agent approach can work in practice, we decided to setup an experimental framework that allowed us to test the agent-based approach vs. the client/server approach from indexing a web site. Two identical machines where used. One was configured as a web server, and the other one as a client. In between, another machine was configured as a router. In this machine, a special program [34] was installed, which allowed the available bandwidth between the two other machines to be controlled. For indexing the web site, we used the indexing package from the Bddbot project [35].

We indexed web sites ranging from 10 Mbytes to 40 Mbytes. It is important to understand that this is the size of the text present on the site, since no other media types are indexed. The chosen bandwidths to test were 64 Kbps, 128 Kbps and 256 Kbps. We believe these bandwidths are representative of the values that many people nowadays get on the internet.

5.2 Results

Figure 4 summarizes the results obtained in the several tests. On the graphs, MA*xxx* refers to the use of mobile agents over a *xxx* Kbps line, and CS*xxx* on the use of client/server over a *xxx* Kbps line. Due to space constraints, we only present the results from 64Kbps and 256Kpbs. These results show that, for this application, mobile agents perform much better than client/server. The performance increase ranges from 70% over client/server when a 256 Kbps setup is used, to 160% when 64 Kbps are being used (Figure 5).

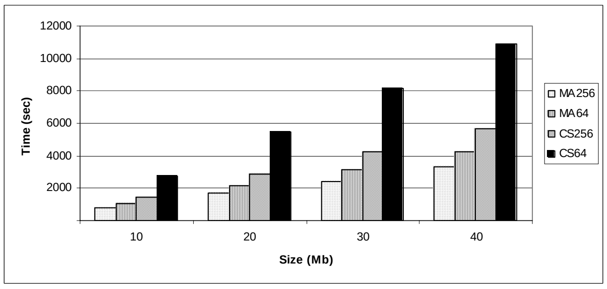

Figure 4 – Client/server vs. mobile agent performance

One interesting point is that the speedup remains more or less constant for a given bandwidth, across all sizes. In fact, this result was to be expected. Because the resulting index is directly proportional to the size of the site, the speedup is directly related to the time it takes to bring the whole web site from the remote location to the local machine (client/server) over the time it takes to just bring the resulting index (mobile agents).

It is also important to note that the speedup curve slightly declines as the size of the web site increases. This is because when the agent migrates to a web site for indexing it, the web server and the agent will be processing on the same machine. This increases the workload of the remote machine. The speedup decline is easier to perceive on larger sites since while the agent is

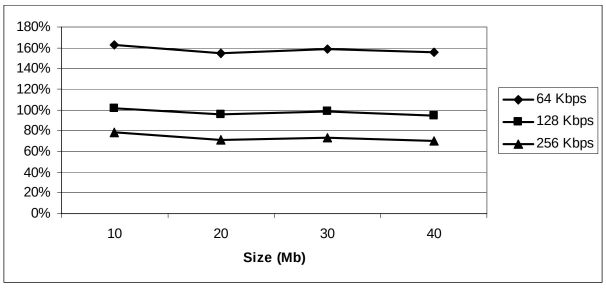

Figure 5 – Performance increase of using agents when compared with client/server

processing it is necessary to create a lot more temporary files, and to do much more intensive processing. Nevertheless, the speedup decline is quite small. The key points to retain from these results are:

- By using mobile agents is possible to obtain large performance increases when compared with a traditional client/server solution.
- Mobile agents can scale well, getting a more or less constant speedup as the size of the work to be done increases.
- If the agents impose a high load on the target machine, the speedup may diminish since only one machine will be being used, instead of two like when a client/server approach is used.

This last point is especially important on the following situation. If a large number of users migrate their agents for performing operations locally on a server, there may exist an important performance degradation. What this means is that when one thinks about deploying an infrastructure as this one, careful planning of resources available on the server and the number of agents that it will be allowed to concurrently execute must be done.

6 Related Work

To our knowledge, existing approaches for integrating mobile agents with the world-wide-web rely on, either building up a mobile agent platform which also supports the HTTP protocol, building up a web server that supports mobile agents, or putting a standard mobile agent platform side-by-side with the web server, but having limited integration.

In [28], Dharap discusses an approach where an agent platform is built, and supports automated browsing of the internet. This platform is able to receive agents that query the local web server, according to its owner parameters, and then forward the agents to another platform. The objective here is simply to allow the agents to access local web information. The agents themselves do not have the capability of publishing information. In [29] a similar approach is described. In this case domain experts implemented as mobile agents, navigate through the web sites, browsing for information. The system is to be built on top of the MOLE mobile agent platfom [30]. In [31], Roth describes an agent platform in which a mobile agent is implemented to act as a web server, and also to allow the execution of servlets. In the case of this project, this agent is static, so it is conceptually identical to build a web server on top of an agent platform.

On [32], Fünfrocken discusses the implementation of a web server that is integrated with an agent platform that is able to receive and execute mobile agents. These agents are able to query and publish information on the web server. As future work the authors indicate that they intend to extend the approach for using it with other web servers. The status of that work is not known at this time. In [33], it is described the implementation of a web server that among other features supports the execution of mobile objects.

To our knowledge, our framework is the only one that is able to integrate with any web server supporting the Servlet Specification, allows agents to query local information, publish information on the site and act as ordinary servlets.

7 Conclusion

In this paper we have presented our experiences on using the M&M framework for developing web 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.
- 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.

Our performance measurements also show that by using a mobile agent approach it is possible to obtain large increases in performance and saved bandwidth.

Finally, 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 discussed 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 FCT, through the M&M project (project reference POSI/33596/CHS1999) and 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] M. Greenberg, J. Byington, and D. Harper, "Mobile Agents and Security," in *IEEE Communications Magazine*, vol. 36(7), pp. 76-85, 1998

[3] W. Farmer, J. Guttman, and V. Swarup, "Security for Mobile Agents: Issues and Requirements," in *Proc. National Information Systems Security Conference*, 1996.

[4] D. Milojicic, "Trend Wars: Mobile Agent Applications," in *IEEE Concurrency*, vol. 7(3), pp. 80-90, 1999.

[5] "Mobile Agent List", available at: *http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole/mal/mal.html*.

[6] Sun Microsystems Inc., "The Java 2 Platform", available at: *http://www.javasoft.com/j2se/*.

[7] F. Hohl, "A Model of Attacks of Malicious Hosts Against Mobile Agents," presented at the *4th Workshop on Mobile Object Systems: Secure Internet Mobile Computations*, France, 1998

[8] T. Sander and C. Tschudin, "Towards Mobile Cryptography," in *Proc. 1998 IEEE Symposium on Security and Privacy*, Oakland, California, 1998

[9] S. Loureiro, "Mobile Code Protection", *Ph.D. Thesis*, ENST Paris / Institut Eurecom, 2001.

[10] F. Hohl, "An approach to solve the problem of malicious hosts," presented at the *4th ECOOP Workshop on Mobility: Secure Internet Mobile Computations*, Brussels, Belgium, July 1998.

[11] 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.

[12] D. Milojicic, B. Breugst, I. Busse, J Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran and J. White, "MASIF, The OMG Mobile Agent System Interoperability Facility," in *Proc. of the Second International Workshop on Mobile Agents (MA' 98)*, Stuttgart, Germany, 1998.

[13] Foundation for Physical Agents Organization, "The specification repository", available at: *http://www.fipa.org/repository/fipa2000.html*.

[14] T. Gschwind, "Comparing Object Oriented Mobile Agent Systems", presented at *6th ECOOP Workshop on Mobile Object Systems: Operating System Support, Security and Programming Languages*, Sophia Antipolis, France, June 2000.

[15] T. Wash, N. Paciorek, and D. Wong, "Security and Reliability in Concordia," in *Proc. of the 31st Annual Hawai International Conference on System Sciences*, Hawai, January 1998

[16] 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

[17] P. Marques, L. Silva, J. Silva, "Security Mechanisms for Using Mobile Agents in Electronic Commerce", in *Proc. of the 18th IEEE Symposium on Reliable Distributed Systems*, Lausanne, Switzerland, October 1999.

[18] "The M&M Project", available at: *http://mm.dei.uc.pt*.

[19] Sun Microsystems Inc., "The Servlet Specification 2.3", available at: *http://www.javasoft.com/servlet*.

[20] Sun Microsystems Inc, "JavaBens Specification 1.01", available at *http://www.javasoft.com/beans*.

[21] 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.

[22] 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.

[23] W3C Consortium, "The Jigsaw web server", available at: *http://www.w3.org/Jigsaw/*.

[24] The Apache Consortium, "The Apache web server", available at: *http://httpd.apache.org/*.

[25] The Apache Consortium, "The Jakarta project", available at: *http://jakarta.apache.org/*.

[26] Alaire Corporation, "The JRUN server", available at: *http://www.jrun.com/Products/JRun/*.

[27] Sun Microsystems Inc, "The Java Web Server", available at: *http://www.sun.com/software/jwebserver/index.html*.

[28] C. Dharap, M. Freeman, "Information Agents for Automated Browsing," in *Proc. of the ACM CIKM'96*, Rockville, USA, 1996.

[29] 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.

[30] J. Baumann, F. Hohl, K. Rothermel, and M. Strasser, "Mole - Concepts of a Mobile Agent System," in the *World Wide Web Journal*, vol. 1(4), 1998.

[31] 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.

[32] 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.

[33] 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.

[34] L. Rizzo, "The Dummynet Project", available at: *http://info.iet.unipi.it/~luigi/ip_dummynet/*.

[35] T. Macinta, "The Bddbot Project", available at: *http://www.endware.com/bddbot/*.