

MOBILE AGENTS: A SUITABLE SUPPORT FOR SERVICE PROVISION IN MOBILE COMPUTING ENVIRONMENTS

OUAHIBA FOUIAL, NADIA BOUKHATEM, ISABELLE DEMEURE

Abstract. This paper aims to investigate the use of mobile agents for service provision in mobile computing environments. A quantitative model and experimental measurements are performed to compare Java RMI and mobile agents in the special context of software downloading. The results show that significant performance improvements can be obtained using mobile agents. In addition, two agent-based services, namely Service Download and Service Discovery, are proposed to illustrate the benefits of using mobile agents for service provision and their potential impacts on mobile computing environments.

1. Introduction

Mobile computing refers to distributed computing systems that enable people to access various distributed network services anyplace and anytime. Mobile computing has become possible with the convergence of mobile communications and computer technologies.

In future mobile computing systems, mobile users will be able to access a large variety of services supported by various network and service providers. They will have the possibility to invoke them via handheld, portable terminals even when moving and with the same facility as on their desktop stations. The main goal of our work is the design of a software platform for service provision in mobile computing environments. This platform will allow the mobile subscribers to download services into their mobile devices which can vary from handheld small devices with limited capabilities (mobile phones, PDAs, ...) to powerful terminals (laptops).

Mobile computing environments can be distinguished from classical wired environments by their very limited, variable and asymmetric bandwidth, frequent and prolonged disconnections, geographical mobility, severe resource restrictions and complex data management [PB94, FZ94]. Unlike their wired counterparts, design of software for these environments should consider all these

constraints. Consequently, new software techniques should be developed and many system integration effects have to be addressed. The design of a software platform for service provision is possibly the most prominent in mobile computing environments. Currently, most distributed systems designed for mobile environments are based on classical communication paradigms like RPC (Remote Procedure Call)[Sri94] and Java RMI (Remote Method Invocation)[jav99]. Mobile agents have received important attention in the last few years as a new programming paradigm of widely distributed and heterogeneous systems.

Commonly, a mobile agent is defined as an independent software program that acts on behalf of a user and helps him to perform tasks. Mobile agents can carry out their tasks autonomously and regardless of the application that dispatched them. This capability, that is at the core of many of the advantages the mobile agents provide, is particularly useful in mobile computing environments where the constraints imposed by the mobile terminals and communication links often force the user to disconnect from the network. In this case, a mobile agent can be unleashed into the network to perform some tasks on behalf of the user, who meanwhile is totally disconnected. Results can be eventually gathered by the user upon reconnection.

The combination of mobile agents and mobile computing environments sounds promising, because it seems that mobile agents could overcome the typical and inherent restrictions of mobile environments. This paper aims to investigate the use of mobile agents for service provision and, in particular, analyses their applicability in software downloading and service discovery. The choice of this concept for service provision is motivated by a quantitative model and a performance evaluation which compare the agent model to the client-server model represented by Java RMI. The results of the evaluation show that significant performance improvements can be obtained using mobile agents for service downloading. Mobile agents perform better when the size of the service to be downloaded is very large and in case of a large number of service providers to be contacted in order to download the best service corresponding to the user's needs.

In addition, two agent-based scenarios are proposed in this paper : Service Download and Service Discovery. The aim is to highlight the benefits of mobile agents, in particular, their efficiency and flexibility in the development of enhanced services.

The remainder of this paper is organised as follows. Section 2 presents the main characteristics of RMI and mobile agents paradigms for software downloading. Section 3 provides a quantitative model that determines the conditions in which the mobile agents outperform RMI. Section 4 presents experimental

measurements and an analysis of obtained results. Section 5 describes some related works regarding the comparison of mobile agents with various distributed communication paradigms. Section 6 presents our work within MOBIVAS¹ project and presents two agent-based services to demonstrate the benefits of mobile agents for service provision. We conclude the paper in Section 7.

2. Java RMI/Mobile Agents - Background

One of the main issues in the development of a comprehensive distributed software platform for service provision is the choice of the technology which should support such a platform while meeting its overall requirements. In the following sections, we will focus on software downloading which is one of the most important aspect of service provision.

Software downloading operation consists of moving services from service providers to the mobile terminal where they can be executed. Thus, the mobile terminal should offer an environment able to support service downloading facilities as well as service execution capabilities.

The first step towards the definition of a downloading environment for mobile terminals was the WAP (Wireless Application Protocol) standard[wap] that enables Internet content and services presented in the web to be accessed by wireless terminals. The main idea of WAP was to address the constraints of a wireless environment and adapt existing Internet technology to meet these constraints. However, possibilities offered by WAP are limited to download textual data and some telephony primitives. WAP can not provide users with multimedia interactive and enhanced services to be executed in their terminals. In this section, we present the main characteristics of mobile agents and Java RMI models. The use of mobile agents and Java RMI for service provision (in mobile communication environments) was widely investigated in a large number of industry sponsored and European ACTS (Advanced Communications Technologies and Services) projects such as Climate[cli], Montage[mon], Amase[ama], Cameleon[cam], . . .

In the following sections, we investigate the use of mobile agents and Java RMI models for software downloading. We derive a quantitative model and carry out experimental measurements to compare both approaches.

It is worth stressing that our platform should allow mobile users to download services on a large variety of mobile devices even on small devices (with limited capacities). This fact is not in contradiction to the choice of Java and

¹MOBIVAS: Downloadable MOBILE Value-Added Services through Software Radio and Switching Integrated Platforms, is an European IST project. URL at: <http://mobivas.cnl.di.uoa.gr>

mobile agents (which are mostly developed in Java) as supports for software downloading. Indeed, it appears clearly that the current trend of both constructors and network providers in mobile communications field is towards equipping mobile devices with a full Java execution environment. This is motivated by the fact that Java offers many interesting advantages such as its standard interfaces (API) and its dynamic behaviour. In the future few years, mobile devices will be probably so much powerful that they will be able to hold a full JVM without any constraint or limit.

2.1. Java RMI

Java RMI is the Java RPC mechanism. It has the advantage over traditional RPC systems that it is a part of Java object oriented approach.

RMI is a simple and a direct model for distributed computation with Java objects. It provides a uniform object designation mechanism hence enabling to invoke a method on a given object, whatever its location, by using a unique global reference.

One of the central and unique feature of RMI is its ability to download the byte code of an object class if the class is not defined in the receiver's virtual machine. The type and the behaviour of an object, previously available only on a simple virtual machine, can be transmitted to another, possibly remote, virtual machine. RMI passes objects by their true types, so the behaviour of those objects is not changed when they are sent to another virtual machine.

The RMI designers extended the concept of class loading to include the loading of classes from FTP servers and HTTP servers. This is a powerful extension as it means that classes can be deployed in one, or only few places, and all nodes in a RMI system will be able to get the proper class files to operate. RMI supports this remote class loading marshalling through the *RMIClassLoader*. If a client or a server is running on a RMI system and sees that it must load a class from a remote location, it calls *RMIClassLoader* to do this task.

Java RMI is a remote procedure call protocol which supports remote class loading but cannot be used solely for software downloading. Indeed, when designing a software downloading system, we would like to have the flexibility to control where a remote object (or service) runs. In the absolute, when a remote object is brought to life on a particular JVM, it will remain on that JVM. It is not possible to send the remote object to another machine for execution at a new location. RMI makes it difficult to have the option of running a service locally or remotely.

The very reason RMI makes it easy to build some distributed applications can

make it difficult to move objects between JVMs. When we declare that an object implements the *java.rmi.Remote* interface, RMI will prevent it from being serialised and sent between JVMs as a parameter. Instead of sending the implementation class for a *java.rmi.Remote* interface, RMI substitutes the stub class. Because this substitution occurs in the RMI internal code, one cannot intercept this operation.

The design of a software downloading system using RMI, needs a manual serialisation of the remote service before its sending to the remote JVM. This requires more coding below RMI level and can lead to extra coding and maintenance.

2.2. Mobile Agents

The mobile agent (MA) concept is a promising and an innovative approach for software downloading support. Commonly, a mobile agent is defined as an independent software program that is able to move from a server to another [FLP98, GKN⁺96]. A Java Applet can be seen as an example of a very limited mobile agent.

The mobile agent paradigm is different from other mobile code paradigms since the associated interactions involve the mobility of the existing computational component. In other words, while in the classical communication paradigms (e.g. remote evaluation and code on demand), the focus is on the transfer of code between components, in the mobile agent paradigm, a whole computational component is moved to a remote site, along with its state, the code it needs, and some resources required to perform the task [Lan98].

The main functionality of a mobile agent system is to provide transparent code migration. Generally, this is supported by a primitive that encapsulates the needed mechanisms for moving the agent code, its data and possibly its state. Most of current Java-based mobile agent systems support weak migration. In this case only the agent code and its data are moved to the remote host to be executed there.

Strong migration implies the transport of the agent code, data, and execution state. This means that an agent can suspend its execution at an arbitrary point, move itself to another machine, and resume execution on the new machine at the exact point at which it left off. Only some mobile agent systems provide this capability because of its subsequent heaviness.

The process behind the agent migration consists in transforming the code of the agent, its data (and its state) into an intermediate format to be transported across the network and restarted on a remote host. In Java based agent systems, the intermediate format is the result of the *Object Serialization* technique. Java

arranges the user's source programs in classes which generate bytecodes at compilation. When the agent travels through the network, its bytecode and those of all objects created or used by it are sent to the remote host. The sender wraps this bytecode in a special structure, by using the object serialization technique. The receiver "deserialises" it to recover the bytecode of the agent and its classes. Before the bytecode is used, a specific ClassLoader is invoked to retrieve the corresponding class.

In most of current mobile agent systems, the agent migration process is encapsulated in a single primitive which provides transparency hiding implementation details from the application developers.

3. A Quantitative Model

In this section, we evaluate in which conditions the mobile agent model outperforms Java RMI model. After a description of the considered scenario, we present in depth the quantitative model and experiment results.

In our scenario, we are interested in evaluating the effect of software downloading on the client system capacity, in particular, the download duration and its implication.

A relevant challenging issue in mobile service provision is enabling the user to locate the desired service between several services offered by a set of servers. In our scenario, we assume that the client wants to retrieve a service proposed by a number of service providers. Each of these service providers offers a specific version of the same service. We assume that each server possesses specific data describing each service it offers.

To be able to retrieve the best version of the desired service, the client has first to retrieve from the servers the data describing each service version. Then, it processes these data in order to select the version which matches best to the user's needs. Once the version is chosen, the client downloads the service.

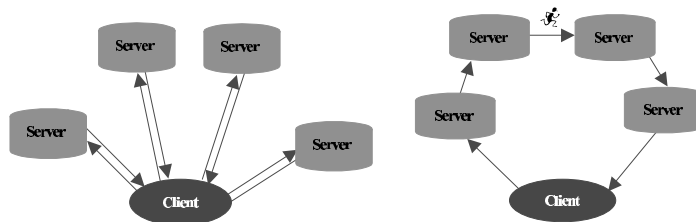


Figure 1: **Java/RMI model versus Mobile Agent Model**

In case of Java RMI model, the client makes a series of remote procedure calls. Each call entails two kinds of messages : the client sends to the server a request which contains the identification of the desired service; the server sends to the client a response which contains the data describing the requested service (figure 1). Thus, assuming that we have n servers, the client should send and receive a total of $2n$ messages.

The mobile agent model only requires sending the message that transports the agent from the client to the first server and then, between the other servers (figure 1). Hence, if n is the number of servers, $(n+1)$ messages are exchanged between all stations.

Our quantitative model is based on two elementary functions. The first one evaluates RMI and MA models in terms of communication cost time, and the second one in terms of data downloading time.

It is worth stressing that the communication cost time is the total time for establishing the communication between the client and a server without considering the time expended for data downloading. These two elementary functions are defined to make easier the formulas elaboration.

3.1. Java RMI Model

Using Java RMI model, each server provides a remote interface which enables the client, providing the name of a service, to obtain the data describing the requested service.

RMI supports two classes that implement a remote interface. The first class, which is the behaviour, runs on the server. The second class acts as a proxy for the remote method and runs on the client.

In order to get the data describing the service to be downloaded, the client should make a remote call to each server. During this call, the client should first download the remote method proxy and then use it to invoke the remote method.

Communication Cost

In RMI model, the time to obtain the final response from one server is the sum of the communication establishment time, and the time of proxy (stub) downloading, and the time of remote method invocation, and the time of the server response.

As the same remote procedure interface is used in each server, the proxy is downloaded to the client only during the first remote call (from the first server). In our formulas, the proxy download duration time is denoted by T_{proxy} and the communication time

between the client and each server ², without the proxy download duration time, is equal to $2t_0$, where t_0 is the time of the establishment of the communication between a client and a server (or between a server and a client).

Considering n servers, the total communication cost time is given by:

$$(1) \quad t_{ComRMI}^n = 2nt_0 + T_{proxy}$$

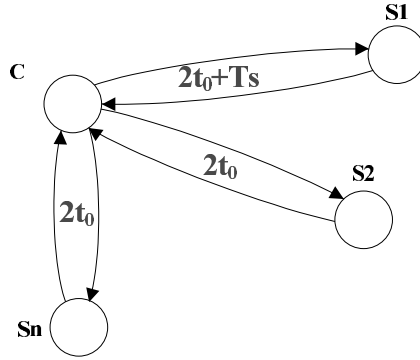


Figure 2: **Communication Cost in Java RMI Model**

Data Downloading Time

The client has to download the same amount of data from each server. The time to obtain the final response from one server is equal to the communication cost time, plus the time t of data downloading ³. Obviously, the time t depends on data size.

Thus, the total data downloading duration is given by:

$$(2) \quad t_{RMI}^n = n(2t_0 + t) + T_{proxy}$$

²The communication cost time is considered the same for all servers contacted by the client.

³We assume that data downloading time is similar whatever the server to be contacted by the client.

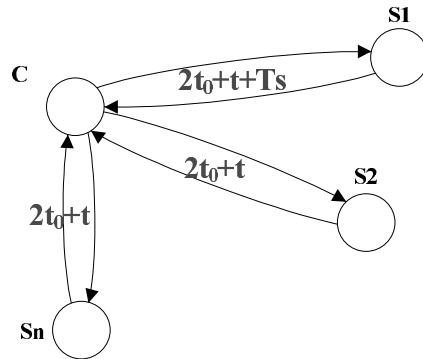


Figure 3: Downloading Time in Java RMI Model

3.2. Mobile Agent Model

Using the mobile agent model, the client sends an agent to negotiate ⁴ with each server in order to retrieve the best service matching the user's needs.

Communication Cost

In this case, we evaluate the time the agent takes to move from a server to another without retrieving the data. This time is given by :

$$(3) \quad t_{ComAM}^n = (n + 1)t'_0$$

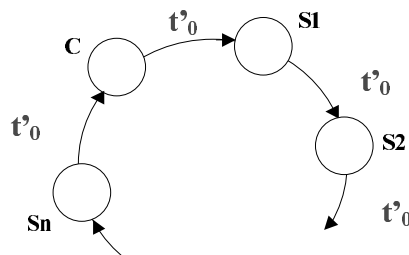


Figure 4: Communication Cost in Mobile Agent Model

⁴The negotiation time between the agent and each server is considered negligible.

where t'_0 represents the agent migration time. To simplify the formulas, we assume that the agent takes the same time to travel from the client to a server, and from a server to another.

Data Downloading Time

At first, let us consider a general case. We assume that the agent moves from a server to another and retrieves the same amount of data from each server. If we note t the download time of this amount of data, then the total downloading time for n servers is given by :

$$(4) \quad t_{AM}^n = t'_0 + (t'_0 + t) + \dots + (t'_0 + nt)$$

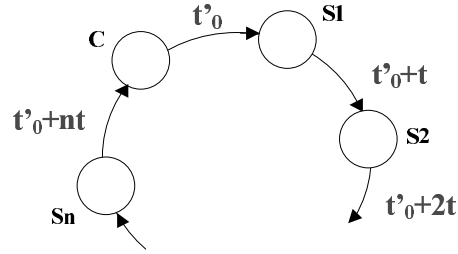


Figure 5: Downloading Time in Mobile Agent Model

$$(5) \quad t_{AM}^n = (n + 1) \left(\frac{nt}{2} + t'_0 \right)$$

In our scenario, the mobile agent has to download the data only once. We assume that the agent transports these data from the last server. The total downloading time is then given by :

$$(6) \quad t_{AM}^n = (n + 1)t'_0 + t$$

3.3. Comparaison between RMI amd mobile agents

In this section, we investigate the relationship between the data downloading time and the number of servers so that MA model is more efficient that RMI model.

The MA model outperforms RMI model, if the data downloading time in case of RMI model is larger that the one in MA model. According to equations 2 and 6, this condition can be expressed by :

$$(7) \quad n(2t_0 + t) + T_{proxy} \geq (n + 1)t'_0 + t$$

We then derive the following relation expressing the number of servers as a function of the data downloading time :

$$(8) \quad n \geq \frac{t + (t'_0 - T_{proxy})}{t + (2t_0 - t'_0)}$$

Note that t'_0 and T_{proxy} are considered as constant and the data downloading time can be expressed as a function of the data size (*DataSize*) and network throughput (*Th*). Equation 8 can then be expressed by :

$$(9) \quad n \geq \frac{DataSize + (t'_0 - T_{proxy}).Th}{DataSize + (2t_0 - t'_0).Th}$$

In the same way, from (6) we derive the following relations expressing the data downloading time as a function of the number of servers:

$$(10) \quad t \geq \frac{(t'_0 - 2t_0)n + (t'_0 - T_{proxy})}{n - 1}$$

$$(11) \quad DataSize \geq \left(\frac{(t'_0 - 2t_0)n + (t'_0 - T_{proxy})}{n - 1} \right) .Th$$

A first analysis of equation 9 shows that the data size turns around a lower limit when the number of servers becomes large. This means that for a given data size which turns around this limit, MA model outperforms RMI model if the number of server becomes large (is beyond a certain value).

4. Experimental measurements and results analysis

A more detailed analysis, based on experimental measurements, is given in the following sections. Our experiments have been performed on a network composed of Sun Ultra-5 5.8 workstations under Solaris 3.6.1 System. The link used is a 100 Mbit/s Ethernet. All experiments described in this paper have been performed with Java environment using JDK1.2 and Grasshopper mobile agents platform (a well-known mobile agent system[gra].

4.1. Quantitative model analysis

The quantitative model described above has been instantiated on the experimental environment to determine the values of the constant coefficients in equation 9 and 11, in particular the values of t_0 and T_{proxy} . Thus, by varying the data size in equation 9, we obtain the number of servers for which MA model outperforms RMI model (figure 7). Figure 6 shows the variation of the number of servers according to equation 11 to determine the data size for which MA model outperforms RMI model.

Results shown in figure 6 and 7 lead to the following remarks:

- For a small amount of data to be downloaded through the network, the MA model needs an important number of servers to perform better than RMI model.
- Conversely, for a small number of servers, mobile agents perform better when the size of data is large. Thus, the more the data size increases, the less the number of servers to be contacted is needed so that MA model outperforms RMI model.
- From a given number of servers and when the data size to be downloaded is around the lower limit, mobile agents are better than RMI whatever the number of servers to be visited.

In the final analysis, we can point out that mobile agents perform better than RMI when either the number of servers or the data size is large.

4.2. RMI/MA performance analysis

In this section, we present the experimental results which are aimed at comparing MA and RMI models for service downloading and at validating the mathematical model described above.

Figures 8, 9, 10 and 11 plot the variation of the downloading duration as a function of the downloaded data size by considering respectively 1, 4, 6 and 9 servers.

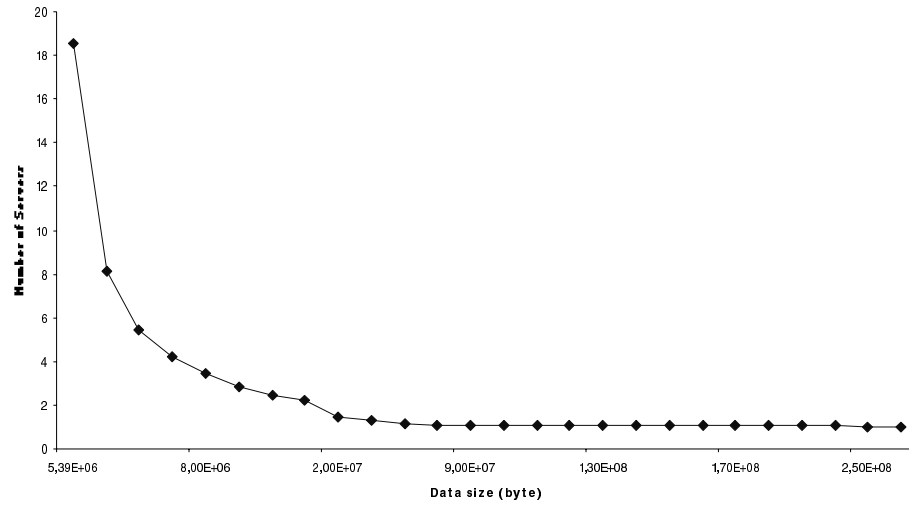


Figure 6: Server Number as function of Data Size (theoretical model)

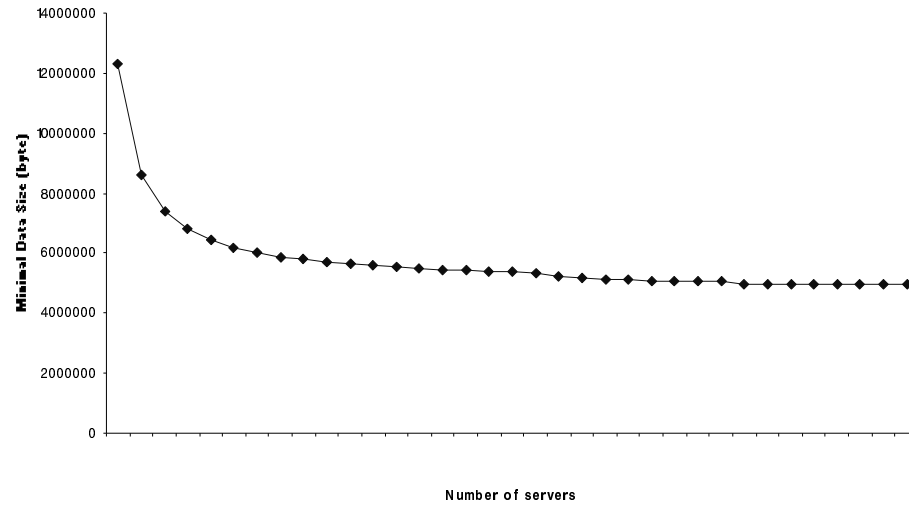


Figure 7: Data Size as function of Server Number(theoretical model)

When considering one server (figure 8), the results show that RMI performs better than MA for both measurements and theoretical results. The reason is

that, in case of RMI model, besides the service describing data to be downloaded, the client also downloads the stub (the proxy of the remote procedure) ; whereas in MA model, the mobile agent code is transported twice between the client and the server. Since, in our scenario, the code size of the mobile agent is much greater than the stub, RMI model outperforms MA model whatever the size of data to be downloaded.

However, in concordance with the mathematical model and according to the experimental environment, when the number of servers is higher than 4, the MA model is more efficient than RMI model if the downloaded data size exceeds a given value (119418 bytes in case of 6 servers). This confirms our conclusions obtained from the mathematical model analysis, in particular, the fact that mobile agents perform better than RMI when either the number of servers or downloaded data size is large.

However, we observe, in all curves, that experimental measurements do not match exactly the theoretical model when the data size is very large. An in depth investigation shows that this is due to the way the operating system manages the file system and data transfer between the memory and the disk (cache management).

In conclusion, we can say that mobile agents make clear improvements over RMI in case of software downloading especially if we are interested in decreasing the response time when downloading an important amount of data through the network, or when visiting a large number of servers.

5. Related work

While many researchers investigated the development of mobile agent systems, very few proposed a quantitative evaluation of this technology through actual measurement in a real large scale environment, along with an evaluation of the basic mechanisms involved in the implementation of such systems.

Authors in [CPV97] give some hints about the process of selecting the suitable communication paradigm for a given distributed application. They consider different communication paradigms namely client-server, remote evaluation, code on demand and, mobile agents. They investigate a simple performance model considering an information retrieval application. The authors argue that there is no best solution to select the suitable communication model when designing a distributed application. Rather, the best solution has to be found on the basis of a careful evaluation of the application requirements against the peculiar features of each paradigm.

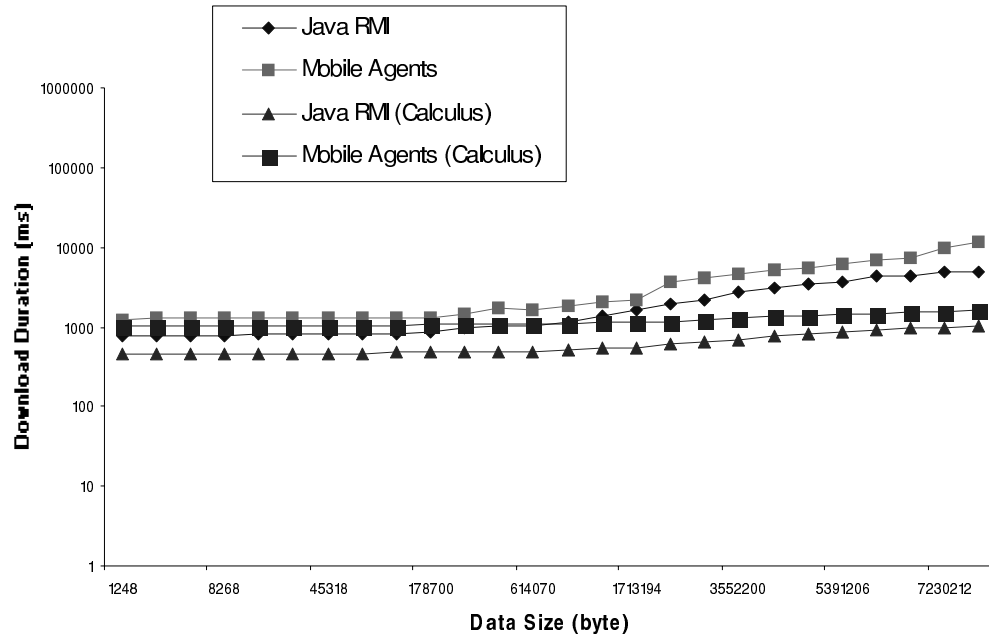


Figure 8: RMI/MA Comparison (1 server)

In [], the authors propose an analytical model that examines the claimed performance benefits of mobile agents over client-server model for retrieving information on behalf of mobile users. They evaluated the model by focusing on the response time as the metric of interest. The results show that, it is not clear that mobile agents are always preferable to client-server or vice versa, in particular, if for a given query, the code size of the mobile agent is larger than client-server request size.

In [IH99], a performance evaluation has been carried out to compare mobile agent and client-server paradigms. This evaluation was performed in Java environment using respectively RMI, the Aglets platform [agl] and a minimal mobile agent platform that the authors developed.

In order to evaluate the interest of mobile agents, two data mining applications were selected. The agents move among different machines in order to look for information on behalf of their users. In one of these applications, the role of the agent is to compress the documents to be transferred in order to reduce the amount of data to be sent. The performance evaluation reveals that the use of

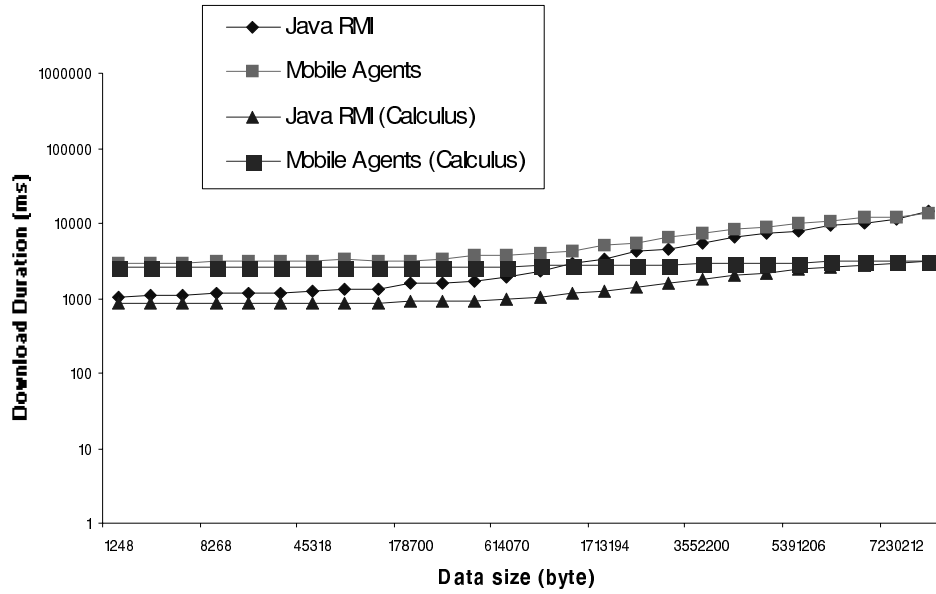


Figure 9: RMI/MA Comparaison (4 servers)

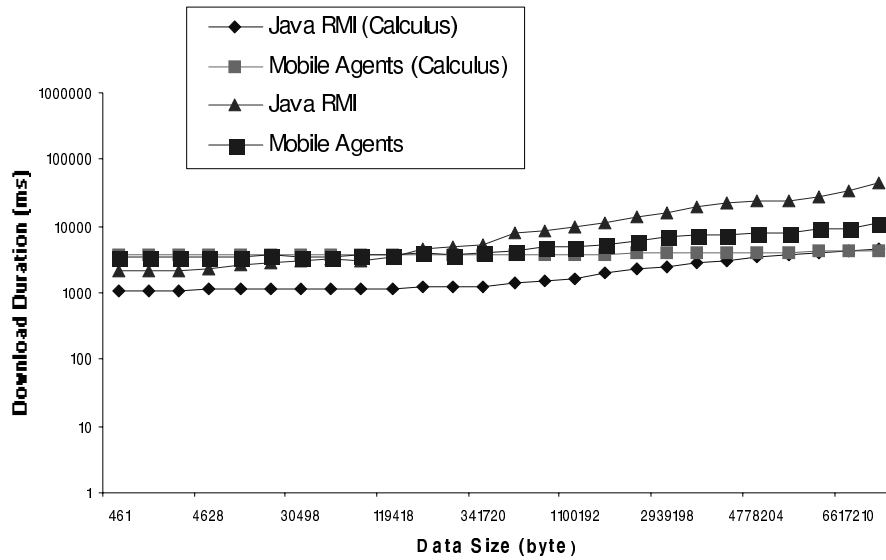


Figure 10: RMI/MA Comparaison (6 servers)

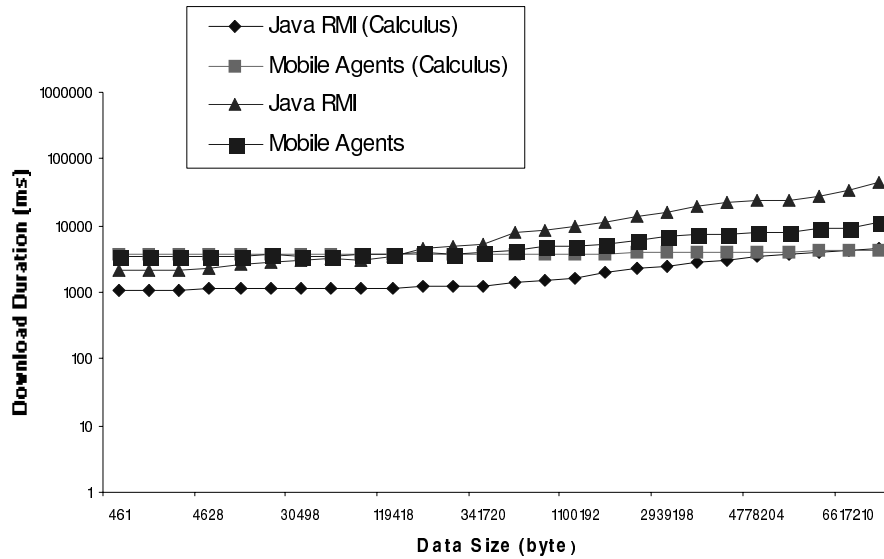


Figure 11: RMI/MA Comparison (9 servers)

mobile agents can lead to a significant improvement especially when the mobile agent, which intensively communicates with a server, moves to that server. This work, as ours, compare Java RMI to mobile agents and confirm our results even the considered applications are different.

The work presented in [OK99] aims to study the possibility of introducing mobile agents to update network nodes MIBs (Management Information Base) in the context of network management domain. A comparative performance study of a client-server and an agent based application is presented. Results of the experimental measurements show that the mobile agent paradigm demonstrates better performance than the client-server when the number of nodes increases. Besides the significant reduction of bandwidth utilisation the agents provide, they carry out additional tasks such as failure detection and routing improvement.

The works presented in this section tied partly our work as far as they compare quantitatively mobile agent and client/server paradigms. However, the performances measured in each work depends tremendously on the type of the application domain for which the study was achieved.

A decision concerning the choice of the suitable support for service downloading based on these works was not sufficient in itself. This leads us to achieve the comparative study between MA and RMI.

6. Agent based service provision

In this section, we present our work within the MOBIVAS project and describe two agent-based services for service provision: Service Downloading and Service Discovery.

Our work was partly motivated by the participation to MOBIVAS project. Our aim within this project is the development of a comprehensive distributed software platform that supports the provision of downloadable services to the subscribers of a mobile communication network.

The software downloading platform that we propose combines two complementary technologies: Java and mobile agents. In such a solution, some communications will be achieved using the facilities provided by Java (e.g. RMI). Service and platform functionality will be implemented basically in Java. The use of mobile agents can complement this solution by facilitating and optimising the development of specific tasks. However, as agent middleware is resource demanding, the agent platform can be installed only in the core network, the service providers and, on some specific terminal classes (PDAs, laptops) where it is realistic to expect enough computational and memory resources to host the mobile agent software platform. Even, as stated previously, in the near future, mobile devices will be probably so much powerful that they will be able to hold a full JVM without any constraint or limit.

6.1. Software downloading scenario

In our model of service provision, a specific component, Service Manager (SM), is defined in the core network to control the downloading procedure between Service Providers and the mobile terminal.

At the core network, a mobile agent platform can provide access to two stationary agents namely the Service Provider Agent (SPA) and the Terminal Service downloading Agent (TSA).

The Service Provider Agent allows the creation of mobile agents which are sent to the Service Providers to download services, handles the incoming mobile agents and interacts with them. This configuration based on stationary agents allows to limit the incoming agents to directly access the resources and hence protect the Service Manager from the malicious agents.

The Terminal Service downloading Agent is responsible for the interaction between the Service Manager and the terminal for service downloading purpose. When receiving a service downloading request (step 1 on figure 12), the Service Provider Agent creates a mobile agent called Mobile Provider Agent (MPA) which contains all the information to identify the service at the Service Provider

level (2). Once it has joined the Service Provider (3), the agent first recovers the service code and then updates its itinerary (4). An itinerary is associated with each mobile agent; it is defined as a sequence of place addresses through which the agent goes to reach its final destination.

As mentioned before, an agent platform is required and deployed into the core network (Service Manager and Service Providers) and possibly in the terminal if the latter has sufficient resources to host the platform.

In case of a resource constrained terminal (mobile phone, for example), the final destination is the Service Manager. Once arrived to the Service Manager (5), the Mobile Provider Agent is taken in charge by the Terminal Service downloading Agent, which uses the RMI capability to download the service to the mobile terminal (7).

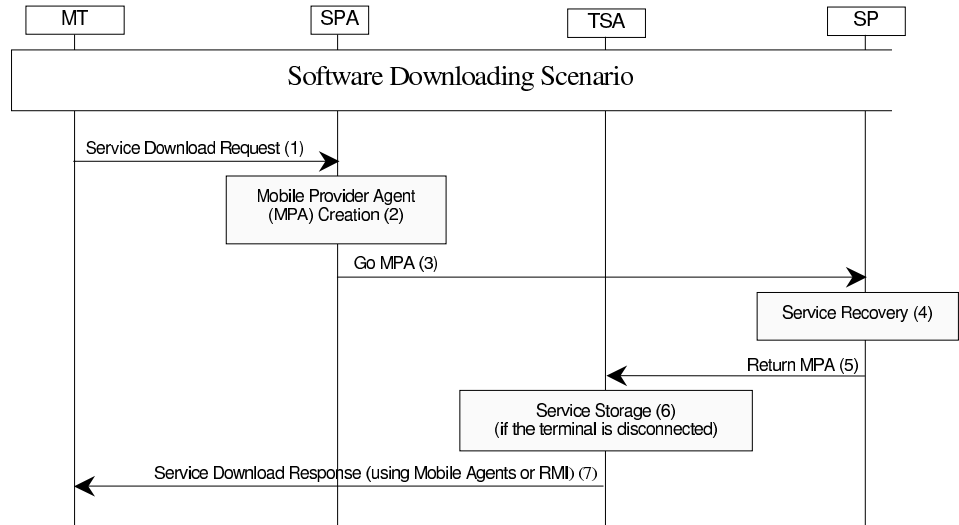


Figure 12: Software Downloading Message Sequence

The Terminal Service downloading Agent handles the downloading between the Service Manager and the mobile terminal and provides several benefits: if the terminal is disconnected, the Terminal Service downloading Agent stores the service in the Service Manager until the terminal reconnects to the network and the downloading process is triggered (6).

In case of a PDA or a Laptop, the final destination of the mobile agent is the

user terminal. The Mobile Provider Agent moves the code from the Service Provider to the mobile terminal (possibly via the Service Manager) (5,7). In this case, the Terminal Service downloading Agent capabilities can also be used if the connection between the Service Manager and the terminal is broken for any reason (6).

This scenario presents several benefits: first, an ongoing interaction between the Service Manager and the Service Provider is not needed since the MA transports all the parameters required for service downloading. In addition, the Service Manager delegates the downloading task to the MA and it does not have to wait for the service provider responses. This mechanism enables to decrease both the load at the Service Manager and the response time in case of large number of service downloading requests are simultaneously sent to the Service Manager.

6.2. Service discovery while roaming scenario

Another field where mobile agents can play a significant role is intelligent information retrieval. A relevant challenging issue in mobile service provision is enabling the user to locate the desired service between possibly thousands of services available from the operator. In the case of roaming, this problem becomes even more complex. The subscriber may need services not available from its home operator (e.g. specific information for the country or city he is currently located). Moreover, it is highly desirable for him to find the services best suited to his needs (in terms of QoS, price, content) among the services offered not just by the serving network, but by a group of operators (e.g. all the operators in the country where he is currently roaming).

The proposed agent-based scenario for addressing this issue is presented in the following.

When a roaming user wishes to access the discovery service, he formulates a request by an appropriate interface. This interface includes an advanced search facility, enabling the specification of detailed requirements, like service description, desirable QoS and price. After the search form is completed, it can be submitted to the serving Service Manager (1) whose the Service Provider Agent creates a mobile agent called Service Discovery Agent (SDA) (2).

This Service Discovery Agent moves consecutively to a number of Service Managers in different networks ⁵ (3,5) and searches for services matching the user request among the services offered by the corresponding operators. In the simplest case, only the home and serving network Service Managers will be

⁵ A prior contractual agreement between mobile operators is required to make this possible.

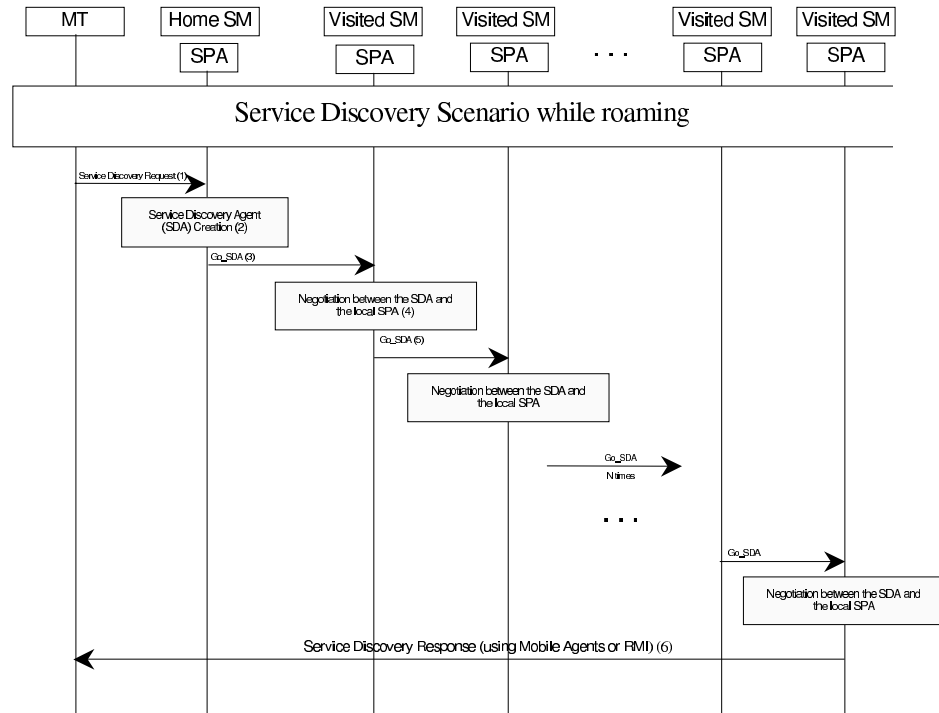


Figure 13: Message Sequence for Service Discovery While Roaming

visited. The agent possesses the intelligence required to perform negotiation with each Service Provider Agent of visited Service Managers and retrieve the best matches to the user’s needs (4). This negotiation could include search refining and intelligent service filtering, according to user preferences and terminal capabilities.

After concluding the search (or potentially only after finding a satisfactory match), the agent returns the results asynchronously to the user (possibly via the Service Manager), in the form of a list of service descriptions and the corresponding links for service access (6).

The above scenario is attractive for several reasons. The most important is that it corresponds exactly to the scenario that we developed in section 3 for evaluating the mobile agent model compared to RMI model and in which we proved that mobile agents outperform Java RMI when either the number of servers or the amount of data to be downloaded is large. Moreover, in this scenario,

the required task is performed asynchronously. There is no need to maintain a network connection over the unreliable and costly wireless link for the whole duration of the task.

7. Conclusion and future work

The work presented in this paper aims to investigate the use of mobile agents in service provision for mobile environments. The mobile agent applicability for software downloading is assessed by several performance measurements which compare RMI and MA models.

The results of this evaluation show that mobile agents provide a significant performance improvement, especially when the number of servers to visit increases or when the amount of data to download is large.

Based on these considerable results and in order to illustrate the possibility of using mobile agents for service provision, two enhanced agent-based services, namely service download and service discovery were proposed.

The work presented in this paper is still under way. Our main perspective is the development of a service provision platform for mobile computing environments based exclusively on mobile agent technology. In this paper, we investigated the use of this technology only for software downloading which represents one of the main requirements of such a platform.

Agent-based developments of the remaining requirements such service customisation, service provision while roaming, user and terminal mobility, and service adaptability and reconfigurability constitute a part of our future work.

References

- [agl] Aglets Agent Platform Homepage. <http://www.trl.ibm.co.jp/aglets>.
- [ama] AMASE Project Homepage. <http://b5www.berkom.de/AMASE/>.
- [cam] CAMELEON Project Homepage. <http://www.comnets.rwth-aachen.de/cameleon/>.
- [cli] CLIMATE ACTS Cluster Homepage. <http://www.fokus.gmd.de/research/cc/ecco/climate/climate.phtml>.
- [CPV97] Antonio Carzaniga, Gian Pietro Picco, and Giovanni Vigna. Designing distributed applications with a mobile code paradigm. In *Proceedings of the 19th International Conference on Software Engineering*, Boston, MA, 1997.
- [FLP98] T. Finin, Y. Labrou, and Y. Peng. Mobile agents can benefit from standards efforts on interagent communication, 1998.
- [FZ94] G. H. Forman and J. Zahorjan. The Challenges of Mobile Computing. In *Computer, IEEE*, 27(4):38–47, April 1994.
- [GKN⁺96] Robert S. Gray, David Kotz, Saurab Nog, Daniela Rus, and George Cybenko. Mobile Agents for Mobile Computing. Technical Report TR96-285, 1996.
- [gra] Grasshopper Agent Platform Homepage. <http://www.grasshopper.de>.
- [IH99] Leila Ismail and Daniel Hagimont. A performance evaluation of the mobile agent paradigm. In *Conference on Object-Oriented*, pages 306–313, 1999.
- [jav99] Java Remote Method Invocation Specification. Revision 1.7. Java 2 SDK, Standard Edition, v 1.3.0, December 1999.
- [Lan98] D. Lange. Mobile Objects and Mobile Agents: The Future of Distributed Computing? In *Proceedings of the European Conference on Object-Oriented Programming, ECOOP'98*, 1998.
- [mon] MONTAGE Project Homepage. <http://www.ccrle.nec.de/berlin/projects/montage.html>.
- [OK99] A. Outtagarts and M. Kadoch. Client-server and mobile agent: performances comparative study in the management of mibs. In *In mata'99, First International Workshop on Mobile Agents for Telecommunication Applications, Ottawa, Canada*, pages 66–81, 1999.
- [PB94] E. Pitoura and B. Bhargava. Building Information Systems for Mobile Environments. In *Third International Conference on Information and Knowledge Management*, 1994.
- [Sri94] R. Srinivasan. Rpc: A Remote Procedure Call Protocol Specification, Version 2, Internet Draft. Technical report, May 1994.
- [wap] Wireless Application Protocol. WAP forum: <http://www.wapforum.org>.

Authors addresses:

Ouahiba Fouial, Nadia Boukhatem, Isabelle Demeure.
Ecole Nationale Supérieure des Télécommunications
Computer Science and Network Department
46, Rue Barrault, 75634 cedex 13
Paris, France
fouial@enst.fr, boukhatem@enst.fr, demeure@enst.fr