Permanent Network Representation for Mobile User

David Carlier[1] and Didier Donsez[2]

1. RD2P - University of Lille 1
CHR Calmette - rue du Pr. J. Leclercq
59037 Lille Cédex - France
e-mail: *carlier@lifl.fr*

2. LIMAV - University of Valenciennes
Le Mont Houy, BP 311
59304 Valenciennes Cédex, France
email: *donsez@univ-valenciennes.fr*

Abstract

The network congestion is currently gowing up in the Internet. This problem is especially important in mobile computing due to low duration of connections. We propose an architecture based on mobile agent concepts. This solution provides a permanent representation of a mobile user on the network even when the user is disconnected. Moreover, the representation is also mobile to be continually close to the user so as to improve the message latency. Java appears to be a language adapted to this model.

Keywords :

Mobile computing, Mobile Agent, Java

1 Introduction

One of the current problems in the Internet is the growing number of users. The connection time gets longer and longer to get data especially during rush hours. This situation is all the more worrying for users of mobile computers. Due to the use of a wireless link, the connection is more and more painful. Due to a weak energy autonomy, the user must frequently switch off his mobile computer or to an idle mode[1], thus gets unreachable from the external world. Several models enables to reduce these drawbacks.

Protocols of communication, flow of communication, time-out are completely different between a wired network and a wireless network. For example, concerning the protocol TCP/IP[2] [S96], time-out is adapted to network having a relatively important throughput with respect to unreliable network with

[1]Idle mode : mode consuming less energy, only the reception of message is available. These features are very common in mobile phones and PDAs.

[2]TCP/IP : Transmission Control Protocol / Internet Protocol

a low throughput such as wireless network. The protocol *VIP*[3] [TUSM94] developed by *Sony CSL*[4] or the protocol developed to the *Columbia University* [IDM91] are adaptation of TCP/IP to mobile communications

Two protocols must be used in a TCP/IP transmission for a communication between fixed stations and a mobile terminal. The I-TCP protocol[5] [BB95] is a protocol developed by Rudgers University for the communication in a cellular network. This protocol allows to use a base station as an intermediary in a communication TCP/IP between a mobile computer and a fixed station.

At the university of *JAIST*, an application of a mobile user is divided into two parts : one on the mobile computer and another one sent to a fixed server [HKN96]. The part on the fixed intermediate station is called a *proxy service*. This *proxy service* allows to filter data destinated to the user, to send pertinent data according to the mobile equipment, thus the size of data circulating through the link between the wired network and the mobile terminal is appreciably reduced. Moreover, processing which requires high computation are delegated to the *service proxy*. The part of the application on the mobile computer and the part on the fixed station are reconfigured each time the user's hardware configuration changes.

Oracle proposes an product named *Oracle Mobile Agent* [Oma95] which allows to create a *representation* of applications destined to mobile users to improve the efficiency and the security of information access. It allows to gather a set of requests in one. This request is sent to a server which executes, then returns results. These actions are asynchronously performed, the user is not forced to be connected during the processing. The client can also initiate a transaction by sending a message to its agent, which carries out the transaction by exchanging a set of messages with the servers and returns results.

1.1 Mobile agent

A *mobile agent* is a software entity able to "travel" throughout a network, to negotiate with other entities (agents or not) so as to achieve a specific task and to reach objectives.

A mobile agent is a new way of organizing the use of distributed resources. A mobile agent can be viewed as an object containing code, data and also a goal and an autonomy. An agent has a task to complete. An agent is therefore created by an initiator, then sent on an acceptor site. The mobile agent is able to *migrate* from an acceptor site to another one then returns the results of the task to its initiator (see the Figure 1).

One of the precursors on mobile agents was *General Magic* with its product named *Telescript* [W94]. However, due to the Java features, most of current solutions are based on Java to take advantages of portabily, checking of the code execution (code interpretation) and ubiquity. The significative current Java mobile agents are *Aglets* [LC96] from *IBM*, *Odyssey* from *General Magic* or *Voyager* [Voyager97] from *ObjectSpace*.

We propose in this paper an architecture to overcome the mobile computing drawbacks : global network latency and frequent user's disconnections. This architecture is based on a combination of two domains : messaging system for mobile computing and mobile agent technology. In our architecture, a mobile agent is associated to a mobile user as his permanent representation on the network. This agent is able to follow the user while he is traveling so as to be continually easily reachable.

Therefore we present in section 2 the concept of user's mobile representation. Then, in section 3, the new programming consequences will be detailed. In section, we will shortly present our implementation choices.

[3] VIP : Virtual Internet Protocol

[4] Sony CSL : Sony Computer Science Laboratory

[5] I-TCP : Indirect - Transfer Control Protocol

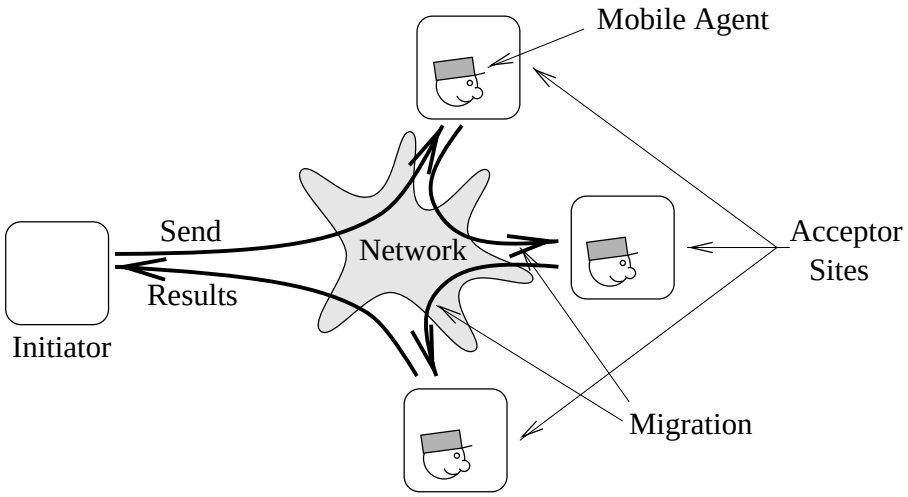

Figure 1: Mobile agent migration

2 User's mobile representation

2.1 Requirement of a mobile representation

In current representations previously described such as O.M.A, mobile applications are represented on the network by a representative task located on a fixed server. This server is full-time connected to the network. However, drawbacks may occur when the user travels to a location far away from the server. On the first hand in such a situation, the user is bothered by a increasing message latency, a reduced throughput. Thus the duration of a user's connection to the network is lengthened. On the other hand, this increases the network congestion.

We propose a different approach to avoid this situation by introducing the concept of mobily of the representation taks. When the user is connecting to the network and his access point to the network is far enough from his representation task, this task gets closer to the user after a migration to a closer server.

We therefore base our system on the mobile agent concept. A mobile representation is associated to a unique user. It is designed as a mobile agent and able to migrate so as to be continually close enough from the associated user. This representation acts as a tasks acceptor. Instead of sending tasks to several servers, all tasks are sent to the same representation. Thus the migration is a global action allowing the move of all tasks at the same time. We call this permanent mobile representation a *Representation Agent* (RA).

2.2 Description of the global architecture

Our architecture is based on a system of *Acceptor sites* (AS). These sites are permanently connected on a global service network (for example, Internet). They supply mobile agents with execution support and a way to migrate from an acceptor site to another one. From an acceptor site, an agent can query local or distant services (Sv) as a classical client in common client-server model. A mobile terminal (MT) is connected to the network via one of the access points (AP) which enables to establish potentially a

connection with any sites on the network. For example, an access point can be the gateway existing between a network GSM and the Internet. However in our software architecture, this terminal contacts the representation agent which is associated to the user of this terminal. This relationship can be stored in a smart card. The smart card allows the user to change easily terminal without a new complex login procedure. It contains data related to the user such as the representation agent location, security features, preference environnement. Thus a user can use any terminal to contact his representation agent. The representation agent is designated to represent continually the user's active applications on the network. This representation agent can *follow* the user when he is traveling. The representation agent migration enables to reduce the distance of the link between the user and his representation agent. The representation agent must consequently find an acceptor site close to the access point and instance a *clone* and migrates the representing tasks of the terminal. However the representation agent always remains active to forward to the clone agent responses and messages destined to terminal's applications.

2.3 The agent's migration

In the example of the figure 2, a user is used to connecting on an access point in the US. Thus he has a representation agent on a close acceptor site. This user can request the services network by requesting the representation agent to instance : (1) a *Software Agent* (SA) to journey on acceptor sites of the network (2,6) to request their local services or distant sites (7). While the reply is not returned, the user can change continent (3) and will be contacted from Japan (4). Its representation agent is then cloned on an acceptor site close to this user (5). The clone agent becomes the main agent until the next move. The initial representation agent acts as a forwarder of responses (8,9) returned by software agents unaware of the migration. In this forwarding mechanism, the initial representation agent permanently exists whereas all clone agents are temporaly created. That is to say, the life time depends on the duration of the user's stay around the access point. When a clone agent is removed, it returns its embedded tasks and their state to the initial representation agent. This mechanism is very similar to the GSM. The user is continually attached to a main center *HLR*[6] where he is registred. However during a trip in another country, he may be also temporaly registred to the *VLR*[7] of a local network.

3 New application design

Current applications are widely based on the Client-server model. This model supposes a reliable connection between the client part of the application and its server part. The designer of client-server applications does not currently take into account the eventuallity of a connection break. Nevertheless, this type of errors is very frequent in mobile computing, thus this model is not adapted to mobile applications. These applications therefore require an other model based on the messaging [ISO93] [L94]. In this model, the different parts of a client-server application send requests and responses to each others during brief connections. The messaging model impies a batch process model of applications. This process is frequent in exchange of document EDI between organizations and enterprises [Y97]. Messaging products use messaging relais which also have functionnalities of switching, conversion and filtering. However, these functionnalities remain limited enough and are not easily extandable and upgradable when new applications are installed on the user's terminal. In the case of a mobile terminal (which may be different during each connection), the relay has to be able :

- to establish a priority among messages destined to the different applications of the mobile terminal,

[6]HLR : Home Location Register
[7]VLR : Visitor Location Register

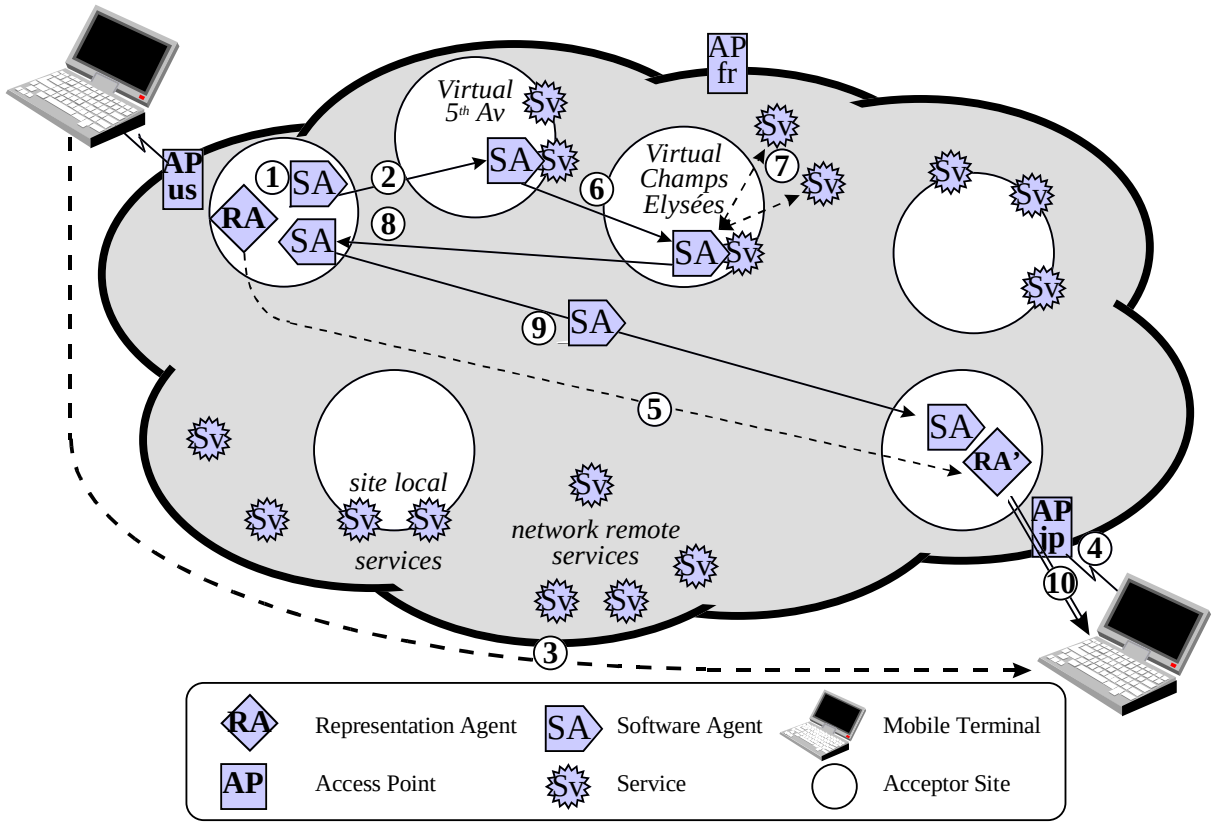

Figure 2: Software agent quest and Representation agent migration

- to adapt the content of a message to the hardware characteristics of the mobile terminal.

For example, an ASCII GSM phone cannot display high definition color image attached to an email. Moreover, an email application commonly displays first only the subject, the date and the sender which may also contain huge attached files. Nevertheless it is only possible if the designer of an application for mobile terminal also programs the "filter/converter" located on the relay. Thus the code of the filter/converter must be as portable and secured as mobile agents. In this context, the representation agent is structured to take in charge :

- the message scheduler[8] according to their priority
- the extraction of components
- the message component conversion according to the mobile terminal features.

The structure of the representation agent embeds several tasks associated to active applications on the mobile terminal. Each task permanently represents the associated application on the network. Its code is composed of parts of code downloaded from both the mobile terminal and code servers distributed on

[8]received messages may be electronic mail or results of request

the network like servlets [JavaServer97]. We better describe the code downloading mechanisms in the article [CT97]. This structure therefore forced the mobile application designer programmer to provide :

- the application code located on the mobile computer
- the representation task code located on the representation agent
- the priorities of messages exchanged between the application agent and the mobile terminal.

Inside the representation agent, tasks upload messages by posting them in priority queues controled by the Message Manager of the Representation Agent (MM_{RepAg}). The queue of higher priority is used for urgent messages as for example the change of a stock value. Posting a message into this queue forces the representation agent to establish a connection to the mobile terminal. Posting into the other queues occurs during a connection with the mobile computer because representation tasks can know the features of the mobile terminal used at this moment. As a similar way, applications on the mobile computer send messages[9] to the Message Manager of the Terminal (MM_{Term}). These exchanges are described on the figure 3.

4 Implementation Choices

A prototype is currently in progress. The Java language [LY97] has been chosen due as the implementation framework for several reasons :

1. Java can be the code used to program a task sent to the representation agent. This code is interpreted and the market acceptance is growing.
2. Java provide facilities of downloading code from a site. Thus tasks can be easily downloaded from a site to another one. The `ClassLoader` and RMI[10] [Rmi97] make easier the code tranferring throughout a network among several Java objects.
3. Java tend to become the base of the most current sofware mobile agent solutions.

Moreover, mobile agent systems such as Aglet Workbench are developping are planned to propose tools to improve security especially to protect acceptor sites.

5 Conclusion

In this paper, we present our solution to reduce connection time of mobile terminal to get services from a network. Our solution is based on a permanent representation of the terminal on the network. This representation is a mobile agent which can follow the user when he is traveling quite far away. The implementation used the Aglet framework which can provide a support to program mobile agents in Java.

However, some problems are not tackled in this paper and in our implementation such as :

1. the way of finding out the closest and the most available acceptor site after a new connection of a mobile computer. A solution can be based on directory.

[9] messages sent by applications of the terminal often are requests

[10] RMI : Remote Methos Invokation

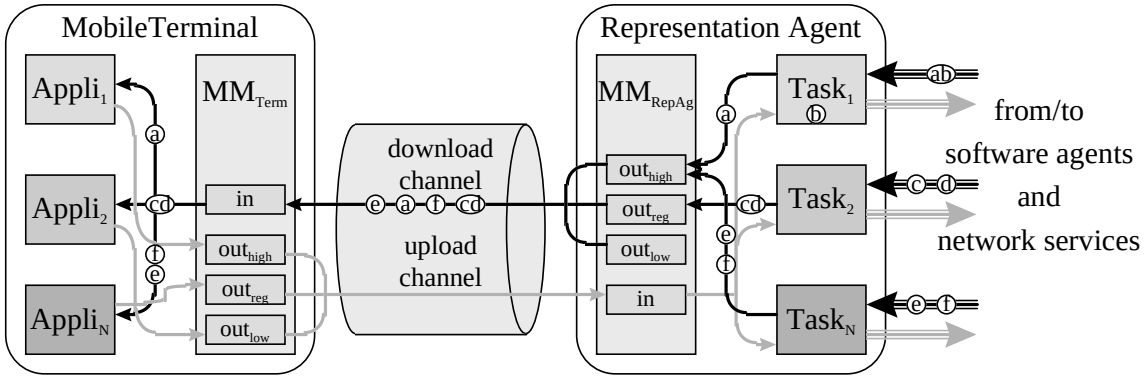

Figure 3: Message exchanges between the mobile terminal and the user's representation agent

2. fault tolerance
   Faulty sites or network misfonction may prevent a representation agent from communicating or being contacted. This problems are partly tackled in [TC96].

References

[BB95] A. Bakre, B.R. Badrinath, *Handoff and Systems Support for Indirect TCP/IP*, 15th International Conference on Distributed Computing systems (ICDCS), May 1995

[CT96] S. Carlier, P. Trane, *Fault-Tolerant Issues Using Agents for Mobile Computing*, in proccedings of second IEEE International On-Line Testing Workshop, pp.182-186 July 1996

[CT97] S. Carlier, P. Trane, *Task Delegation Model Assigned to Mobile computing*, in proccedings of the First International Conference on Information, Communications and

Signal Processing IEEE-ICICS'97, Vol.1, pp.220-224, September 1997

[HKN96] A. Hokimoto, K. Kurihara, T. Nakajima, *An Approach for Constructing Mobile Applications using Service Proxies*, IEEE 16th International Conference on Distributed Computing Systems (ICDCS'96), May 1996

[IDM91] J. Ioannidis, D. Duchamp, G.Q. Maguire, *IP-based Protocols for Mobile Internetworking*, in Proceedings of ACM SIGCOMM, pp.235-245, September 1991

[ISO93] ISO International Standards Organization, *Information Processing Systems - Open Systems Interconnection - OSI TP Message Queueing*, SIO/IEC CD 10026-7 1993(E), 1993.

[JavaServer97] *JavaServer : Developer Documentation*, Sun Microsystems Inc., 1997

[L94] S. Lecomte, *KIOTO: Objet nomade et Travail coopératif*, Mémoire de DEA, LIFL, University of Lille I, July 1994.

[LC96] D.B. Lange, D.T. Chang, *IBM Aglets Workbench, Programming Mobile Agents in Java*, White paper, Tokyo Research Lab., IBM Corporation, September 1996

[LY97] T. Lindholm, F. Yellin, *The Java Virtual Machine Specification*, Sun Microsystems Inc., Addison-Wesley Editions, 1997

[Oma95] *Oracle mobile Agents*, White paper, Oracle, 1995

[Rmi97] *Remote Method Invocation Specification, Revision 1.4, JDK 1.1*, Sun Microsystems Inc., February 1997

[S96] W.R. Stevens, *TCP/IP illustré, Les protocoles*, Volume 1, International Thomson Publishing France, 1996

[TC96] P. Trane, D. Carlier, *Diagnosis Algorithm for Mobility-Oriented System*, IEEE-ASAP'96 : International Conference on Application Specific Systems, Architectures and Processors, August 1996

[TUSM94] F. Teraoka, K. Uehara, H. Sunahara, J. Murai, *VIP : A Protocol Providing Host Mobility*, Communications of the ACM 37(8), August 1994

[Voyager97] *Voyager : Core Package Technical Overview*, Rapport technique, ObjectSpace Inc., Juillet 1997

[W94] J. White, *Telescript Technology : The Foundation for the Electronic Marketplace*, White paper, General Magic, 1994

[Y97] YEKA Editions, *Guide Pratique de l'EDI*, ISBN 2-7337-0078-2, June 1997