Mobile Networks and Applications 7, 315–328, 2002

Keep Your Data Safe and Available While Roaming

YOLANDA VILLATE
CSSI Department, University of Zaragoza, Maria de Luna 3, 50018 Zaragoza, Spain

ARANTZA ILLARRAMENDI
CLS Department, University of the Basque Country, Apdo. 649, 20080 San Sebastian, Spain

EVAGGELIA PITOURA
CS Department, University of Ioannina, GR-45110 Ioannina, Greece

Abstract. The possibility of accessing and/or receiving local or remote data anywhere and at anytime constitutes an important advantage in many business environments. However, when working with mobile devices, users face many problems, such as: (1) *devise exposure problems* – mobile devices are more vulnerable and fragile than stationary devices, because they can be easily stolen, lost or damaged, (2) *media problems* – wireless communications are often unstable, asymmetric and expensive, and (3) *availability problems* – mobile devices stay disconnected for long periods of time. To alleviate these problems, we present a service, *Data Lockers*, which offers to its users first, the possibility of keeping their data in a secure and safe space in a proxy, thus alleviating the device exposure problem. Next, data stored in a data locker are available even when the mobile device is disconnected, thus providing a solution to the availability problem. Finally, specific tasks are carried out at the fixed network on behalf of the mobile user, in this way relieving the media problem. The architecture of the *Locker Rental Service* is based on mobile agents. These agents, and the locker, stay always close to the location of the user, traveling to meet the user wherever the user moves, therefore, allowing users to use anywhere-anytime, ubiquitous persistent storage space located at the fixed network.

Keywords: mobile computing, multi-agents systems, data storage, wireless services

1. Introduction

The sheer rapidity of the spread of both wireless and Internet technologies are favoring a new telecommunication revolution. In the near future, the general-purpose devices of today will be complemented with new easy-to-use, low-maintenance, portable, ubiquitous, and ultra reliable task-specific devices [4]. Even if future devices may not be as resource-limited, they will still be constrained in terms of size, power consumption, and intermittent connectivity. To confront such constraints, it is desirable to define a framework for mobile computing where services can be accessed, or tasks can be accomplished, ubiquitously and without the need of the direct intervention of the user. To this end, we are developing the ANTARCTICA[1] System [7] that provides a set of data services that enhance the capabilities of Mobile Units (MU) and offers new possibilities to mobile users. The architecture of ANTARCTICA is based on the *Client/Intercept/Server* model [18,19] that incorporates modules and agents both at the mobile devices and at intermediary elements, or proxies, situated at the fixed network. Each proxy is called a *Gateway Support Node*[2] (GSN).

[1] ANTARCTICA: Autonomous ageNT bAsed aRChitecture for cusTomized mobIle Computing Assistance.

[2] The Gateway Support Node name is borrowed from the General Packet Radio Service (GPRS). We take GSM and GPRS as cellular network model for our work.

The goal of this paper is to describe the design and implementation of a service central to ANTARCTICA: the *Locker Rental Service*. This service incorporates mechanisms that allow mobile users to rent storage space, called *lockers*, at the GSN. The service provides a variety of types of lockers so users can choose the one that better fits their needs and preferences. The *Locker Rental Service* offers several advantages to the users of mobile units:

1. *Storage space.* A mobile client can deposit in the locker all required data and results obtained for it. The locker constitutes an ubiquitous and persistent storage space kept always close to the user, available at anytime and from anywhere.
2. *Data protection.* The data stored in the locker are protected against unauthorized accesses and modifications as well as any unexpected failures. This characteristic makes the locker an appealing option for maintaining a secure backup copy of some files of the MU, such as configuration files or files containing sensitive information, that need to be protected against losses.
3. *Higher presence at a lower cost.* An agent manages autonomously the locker contracted by the mobile client, so the user can stay disconnected for longer periods of time. The agent can retrieve and manipulate data stored in the locker or store new data in it.
4. *Wireless communications optimizations.* The space in the locker can be used to store data until it is possible or de-

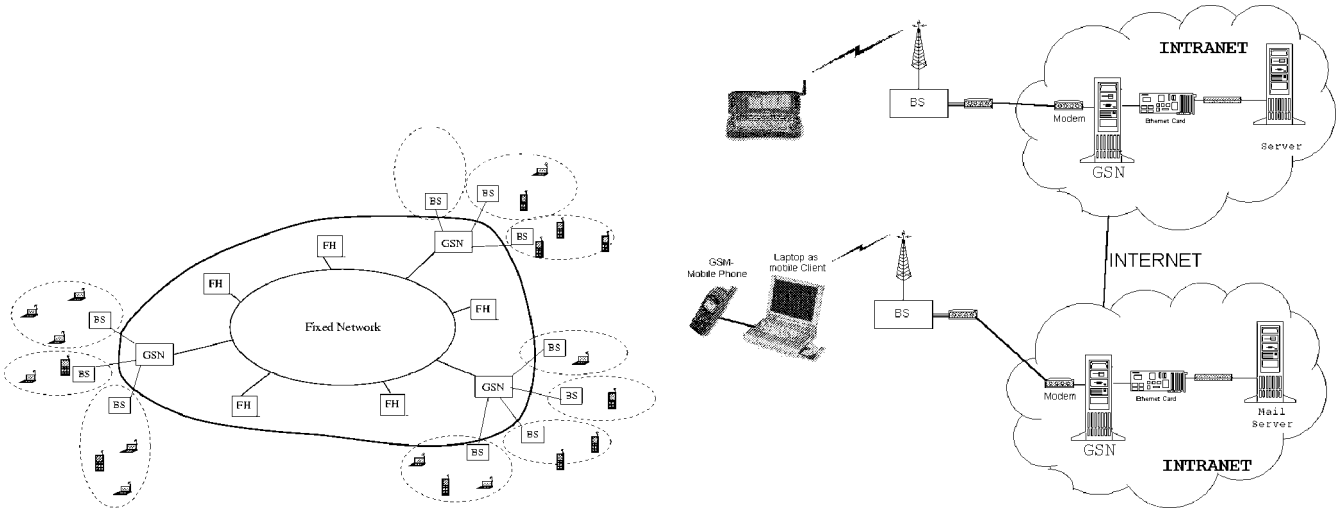

Figure 1. (a) GSN owned by a cellular phone company. (b) GSN owned by a particular company.

sirable to send them to the MU. Before sending data to the MU, the data can be preprocessed, filtered or adapted to the needs, preferences and characteristics of the mobile unit and its user.

5. *Flexibility and adaptability.* The implementation of the *Locker Rental Service* using mobile agents provides for the dynamic adjustment of the space allocated to each locker. It also provides for the easy adaptation of the locker, by facilitating the dynamic creation of lockers tailored to the users needs. Furthermore, it physically supports mobility of the locker to follow the movements of the MUs.

The rest of this paper is structured as follows. First, we introduce the *Locker Rental Service* framework and the different types of lockers. In section 3, we elaborate on the different agents involved in the *Locker Rental Service*, while in section 4, we present some advanced features of the service. In section 5, we study the mobility of the lockers. In section 6, we report on our implementation and present performance results. Finally, we compare our system with related works and offer our conclusions and plans for future work.

2. The Locker Rental Service framework

The *Locker Rental Service* is offered within the ANTARCTICA system. The architecture of ANTARCTICA is based on two pillars: the software agents technology[3] and the *Client-Intercept-Server* (CIS) model. At the fixed network, GSNs (or proxies) act as intermediary elements in the communication between the mobile units under their coverage and all other hosts in the network, mobile (MU) or fixed (FH). The GSN controls the mobile units under its coverage (e.g., their identification, profiles, or groups in which they may be included).

[3] Following the *Mobile Agent System Interoperability Facilities Specification* (MASIF), an *agent* is a computer program that acts autonomously on behalf of a person or organization and a *place* is a context where an agent can execute [16].

The GSNs can be situated at different locations. Figure 1(a) depicts the widely accepted architecture for mobile computing [10] extended with our proposal of supporting GSNs. Notice that in the previous scenario the GSNs are always placed at the fixed network. The number of the GSNs depends on the number of users. In a different scenario in which the services are offered inside a company to its own mobile workers, the ANTARCTICA system can be supported by computers situated at the fixed network but in the company intranet (see figure 1(b)). The concepts involved in the *Locker Rental Service* are valid in both scenarios, but in this paper we take the first one as a reference.

In the GSNs, there is a place called *Inventory* where a particular type of agents, called *majordomo* agents, are executed and, where they can get the information they need about other GSNs, places and services in the system. In the ANTARCTICA System, each one of the *majordomo* agent represents and works for a mobile user. Besides the *Inventory*, there are specialist places, such as for example the *Locker Place*, which is the one related to the *Locker Rental Service*. The ANTARCTICA system allows the user to download files, access databases, navigate the Internet, download and use software. However, the *majordomo* agent representing the user in the GSN can not keep with it large amounts of data for long periods of time. Consequently, the user often has to reconnect in order to download data to the MU and relieve the *majordomo* from the load. To avoid frequent reconnections, the user contracts the *Locker Rental Service*. The *Locker Rental Service* gives the user the possibility of occupying a locker, or vacating it, as needed. Figure 2 shows the elements involved in the *Locker Rental Service*: agents, mobile units and GSNs.

Lockers belong to categories. Lockers of different categories vary in the maximum size they can reach, the services they include and their price. In addition, the *Locker Rental Service* provides two basic kinds of lockers: private and shared lockers. A *private locker* is related to a single user; the data stored in it belongs to that particular user and

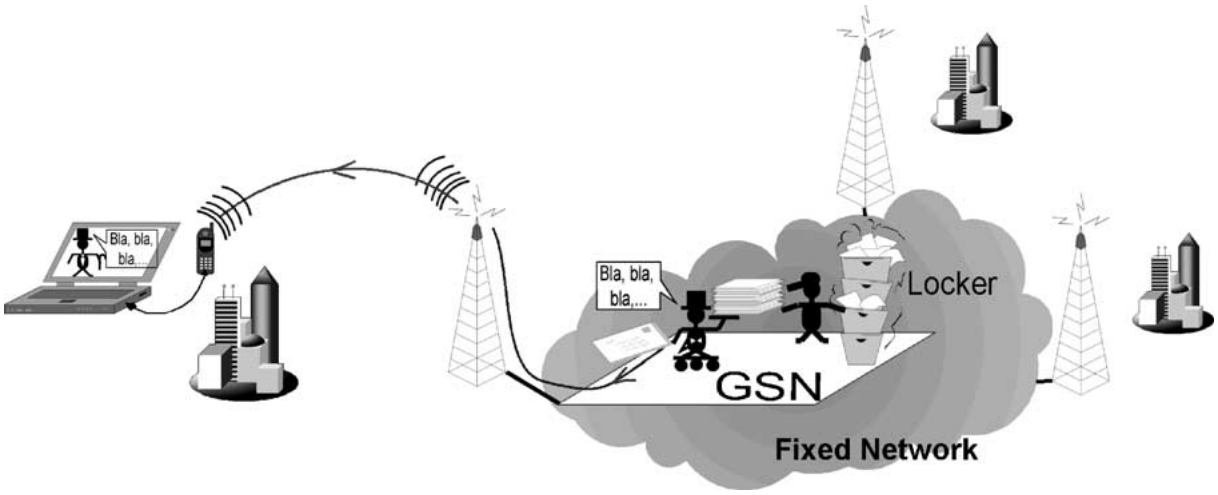

Figure 2. Intuitive representation of the *Locker Rental Service* at the ANTARCTICA system.

can be accessed or modified only by authorized agents representing the user. By contracting a private locker, the user gets the privilege of using this service. During the process of contracting a private locker, the following are specified to determine access control and charging for the requested locker:

(1) an access key or password for the locker;

(2) the range of locker sizes that the user can occupy;

(3) the category of the locker;

(4) the user's privileges;

(5) the rate by which the user is charged for renting a locker with the above characteristics.

A *shared locker* is a locker rented by a group of users. A shared locker distinguishes between data to be used by all users in the group and data for each particular user, by managing sub-lockers for each of them. So, shared lockers are able to store both data private to each user in the group, and data to be shared by all users in the group as well as protecting such data from unauthorized accesses. To contract a shared locker, besides the specifications described above for obtaining a private locker, the following information must also be provided:

(1) the group of users that can use the locker;

(2) the maximum number of users allowed to access the locker concurrently;

(3) the size of the common area of the locker and of the particular sub-lockers for each user;

(4) the services to be provided by the shared locker such as blackboards and meeting places.

A private locker is created when a user asks to use the service and is destroyed when the user decides to vacate the locker. A shared locker is created when the contract is signed and persists for the whole duration of the contract. That is, in contrast to a private locker, a shared locker exists even when no member of the group is using it. A user may have one private and several shared lockers.

3. Agents involved in the Locker Rental Service

In this section we present the agents that participate in the *Locker Rental Service*. Two main advantages of software agents, and in particular of mobile agents, justify our choice of this technology. The first advantage is a *tactical* one, mobile agents give the means to obtain better performance by:

(1) allowing to reduce the traffic of data in the wireless network, between the MU and the GSN;

(2) using local resources (e.g., disk space) offered by the GSN and other fixed hosts;

(3) providing support for disconnected operations (asynchronous communications); and

(4) facilitating an efficient management of data transfer interruptions.

Moreover, the flexibility and adaptability of the implementation of the *Locker Rental Service* using agents facilitates the dynamic management of storage, the adaptation of the locker, and the physical mobility of the locker following their users. The second advantage is a *strategic* one, agents can be customized according to the task they must develop, the preferences of the user they represent, the actual state and characteristics of the computer and the network used [6,15]. While there is a small overhead in using agents, due to the interchange of messages among them, this cost is compensated by the advantages mentioned previously.

Five different kinds of agents are involved in the *Locker Rental Service* (see figure 3) as well as the following places: the *MU Place* in the mobile unit, and the *Inventory Place* and the *Locker Place* in the GSN.

Each specific kind of agent has a different specialization area and mission to perform, for which it may need to interact with other agents and places. However, all agents are designed using the same pattern structure composed by the following basic modules:

(1) the *Agent Communication Module*, in charge of the semantic level of the communications among agents;

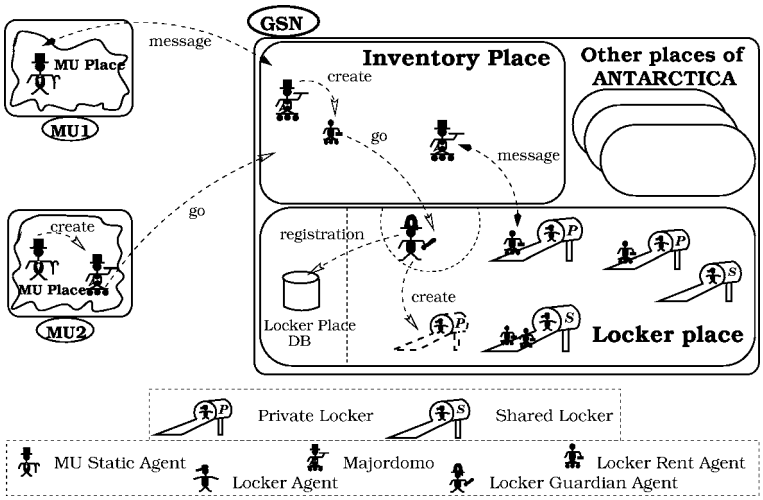

Figure 3. Elements involved in the *Locker Rental Service*.

(2) the *Mobility Module*, that implements the agent mobility policy, controlling when and where to move;

(3) the *Agent Creation and Interaction Module*, in charge of the creation and management of other specialist agents;

(4) the *Specific Knowledge Module*, that contains and manages the knowledge that the agent needs in order to achieve its goal; and

(5) the *Graphical User Interface (GUI) Generator Module*, in charge of the dynamic creation and management of GUIs of the agents that need to interact with the mobile user who owns the MU.

Each kind of agent specializes only the modules it needs, that is not every agent has all the modules.

3.1. Agent features and knowledge

We present the main features and the knowledge managed by the five different kind of agents involved in the *Locker Rental Service*: the *MU static* agent, the *majordomo* agent, the *locker rent* agent, the *locker* agent, and the *locker guardian* agent, shown in figure 3.

3.1.1. The MU static agent

At each Mobile Unit there is a static agent running at any time that the MU is active. This static agent isolates the user's applications from network availability and the specific characteristics of the MU, and is responsible for the administration of the MU's resources. In addition, this static agent creates and launches a single agent from the MU to the fixed network, that is called the *majordomo agent* of the mobile user.

The static agent of the MU is composed through the specialization of four of the basic modules: the *Agent Communication Module* to manage the communications with the *majordomo* agent; the *Agent Creation and Interaction Module* to create and collaborate with the *majordomo* agent; the *GUIs Generator Module* to generate and manage the GUI used to communicate with the user; and the *Specific Knowledge Module* to contain and manage the following kind of knowledge related to its specific role in the system:

(1) the MU characteristics and limitations, resources available and their current state, software installed and so on;

(2) the user profile and preferences;

(3) the wireless communications characteristics and optimization mechanisms;

(4) historical traces of the agent interactions with the user (i.e., services required by the user, etc.), and with the *majordomo* agent; and

(5) a local cache maintenance and management.

This knowledge allows the static agent at the MU to learn, foresee, take decisions, help the user, and abstract the user and the user applications from the network availability and the mobile device features, as much as possible.

3.1.2. The majordomo agent

The *majordomo* agent is created at the MU by the static agent and launched to a GSN at the fixed network. In each GSN, there is a place called *Inventory* where *majordomo* agents execute and can get the information they need about other GSNs, places and services in the system. Once launched, the *majordomo* agent remains in the GSN for the time period required, working on behalf of the MU even while the MU is disconnected. Each mobile computer has its own *majordomo* with the aim of providing adequate services to its owner. Communications of the MU with its outside world are surveyed by this pair of agents: the static agent of the MU and the *majordomo* agent in the GSN. These two agents work together in order to adapt and optimize the communications and use of the wireless media by the MU.

The *majordomo* agent can create other specialist agents to work for it and distributes and coordinates the tasks among these servants agents, in order to fulfill the requests of its user. The *majordomo* agent is composed by the specialization of

the *Agent Communication Module* in order to communicate with its creator, the static agent of the MU, and with the specialist agents it creates; the *Agent Creation and Interaction Module* to create specialist agents that help it accessing the services provided by the system and requested by the user; the *Mobility Module* to support its mobility between the MU and the GSN, as well as among different GSNs in order to follow the movements of its MU; the *Specific Knowledge Module* to contain and manage the following kind of knowledge related to its specific role in the system:

(1) the MU characteristics and limitations, resources available, software installed and so on, to consider when fetching, adapting and sending data to the MU (i.e., color screen);

(2) the actual user preferences and profile, and traces of the user's behavior and interactions with the ANTARCTICA system;

(3) the wireless communications characteristics and resources (i.e., SMS service), and optimization techniques;

(4) the status of the specialist agents working for it, the tasks they are developing, localization, and how to distribute tasks among them to coordinate their work; and

(5) historical traces of the user requests, network status, MU movements, services in use, requests in process, tasks to perform and so on.

The previous knowledge allows the *majordomo* agent to learn in order to work autonomously without asking for the intervention of the user, to filter the data obtained according to the user preferences and the MU capabilities, and to optimize the use of the wireless communications, while representing the user in the fixed network.

3.1.3. The pair of locker agents

When the *majordomo* needs to use the *Locker Rental Service*, it creates an agent called *locker rent agent* and sends it to a *Locker Place*. The *Locker Place* is the specialist place in the GSN that offers the *Locker Rental Service*. Lockers are implemented using a new kind of static agents called *locker agents*. The *locker agents* are created by the *guardian agent* in the *Locker Place*. Each one of them is assigned to a specific user or group of users, i.e., to their *locker rent agents*. This pair, the *locker agent* and the *locker rent agent(s)*, constitutes the locker and takes care of storing the user's data, saving e-mail messages, processing results and communicating with the MU's *majordomo* agent. The fact that both agents have to communicate with each other and interchange data incurs some overhead. However, this interaction is local as both agents reside in the same place, the *Locker Place*. Furthermore, by having both kinds of agents, we can take advantage of specializing them.

The *locker agent* is specialized in interacting with the place and the available resources in order to store and recover data, and assure the privacy and security of its locker. Moreover, the *locker agent* of a shared locker interacts with a group of *locker rent agents*, acting as an intermediary. It also handles concurrent data accesses. The *locker agent* is composed by the specialization of the *Agent Communication Module* in order to communicate with its creator, the *locker guardian agent*, and with the *locker rent agents* it is assigned to serve; and, the *Specific Knowledge Module* to contain and manage the following kind of knowledge related to its specific role in the system:

(1) the *Locker Place* characteristics, resources and facilities available;

(2) the features of the *locker rent agents* and users it is serving;

(3) the user or group contract characteristics and limitations for the use of the *Locker Rental Service*, and of the actual occupied locker, such as space limit and services included;

(4) historical traces of the locker content and size, and its current state;

(5) the locker related services and features (i.e., blackboard, filter library, e-mail redirection, etc); and

(6) the security and accounting policies to enforce.

The *locker rent agent* is specialized on interacting with the user's *majordomo* agent. It has the knowledge about the user, as well as semantic knowledge about the data and how to utilize them. The *locker rent agent* uses the services that the *locker agent* provides to it to store and recover data. It also communicates and shares information with the *locker rent agents* of other users sharing the locker to fulfill the needs of its user. For example, the *locker agent* knows how to store e-mails in the locker, but the *locker rent agent* knows how to ask for them and compose a summary list with the new e-mails received, specifying the date, subject and sender. That list is sent to the user so that the user can choose which e-mails to read. The *locker agent* allows the *locker rent agent* to write or read the blackboard, but the one that knows how to interpret the messages or what to do with them is the *locker rent agent*.

The *locker rent agent* is composed of the specialization of the *Agent Communication Module* in order to communicate with its creator, the *majordomo* agent, with its partner the *locker agent* and with the *locker guardian agent*; the *Mobility Module* to support its mobility between the *Inventory Place* and the *Locker Place*, and among *Locker Places* in different GSNs in order to follow the movements of its MU; the *Specific Knowledge Module* to contain and manage the following kind of knowledge related to its specific role in the system:

(1) the user preferences, and some relevant characteristics of the MU;

(2) how to use locker services and facilities available;

(3) historical traces of the interactions of this agent with other agents;

(4) basic results processing skills (i.e., e-mail filtering); and

(5) policies to enforce and secure its integrity and its interactions with other agents and the user.

The different kinds of lockers are built using different subspecializations of the modules of both the *locker rent agent* and the *locker agent*.

3.1.4. The locker guardian agent

The *Locker Place* offers disk space for renting, so it has mechanisms to administer and monitor the available space, to assign spaces, and to register data necessary for billing for the service. For the surveillance of the use of the service and its resources, the *Locker Place* has associated a database and a *guardian agent*.

When the *Locker Place* is created a static agent is created within it, the *locker guardian* agent. The *locker guardian agent* is always active, its mission is to monitor the population of agents in the *Locker Place*, check the authorization and authentication of incoming agents, create and dispose *locker agents*, maintain a register of the agents in the database (DB) and monitor the use of the resources. The *locker guardian agent* is composed by the specialization of the *Agent Communication Module* in order to communicate with the *locker rent agents* and *locker agents* that populate the place; the *Agent Creation and Interaction Module* to create and manage the *locker agents* it has to create; and the *Specific Knowledge Module* to contain and manage the following kind of knowledge related to its specific role in the system:

(1) the *Locker Place* resources, policies established in the system for their use and how to enforce the policies and optimize the use of the resources;

(2) the performance monitoring, optimization techniques and heuristics, load balancing, and so on;

(3) historical traces of the use of the service, resources and facilities, made by the agents and users of the system, the load of the system, performance and so on;

(4) how and when to interact with other system places, services, agents and resources; and

(5) the security policies to enforce.

The DB maintained in the *Locker Place* registers information about the agents populating it, the users they represent and the data kept in the lockers. In that way, it helps controlling the population and contents of the lockers in a persistent way, and can be used to automate the process of looking for inconsistencies or fraudulent uses as well as abuses (see figure 4).

3.2. UML description of agents behavior when processing tasks

Agents communicate with each other by interchanging messages, this function is performed by the *Agent Communication Module* of each agent. There is an interface, or set of messages, that an agent can understand and that it can send to other agents. Agents interpret the messages to find out the task they are being asked to perform or other information that is conveyed to them.

In figure 5 we present the interaction diagram, described using UML, that shows the suite of messages that agents and the *Locker Place* interchange during the creation of a private locker. Upon receiving a message ordering it to "occupy a locker", the *majordomo* creates the *locker rent agent* and provides it with the data it needs in order to occupy a locker having the characteristics that the user has specified, as well as with the list of places where it can attempt to occupy the locker, or *itinerary*. When a *locker rent agent* is detected arriving to the *Locker Place*, the *locker guardian agent* interrogates and examines the incoming agent in order to check the authorization and authenticity of the agent and its user, and decides whether to allow the agent to occupy a locker or not. If yes, the *guardian* creates a *locker agent* and introduces the two agents to each other so they can work together. Then, the *locker rent agent* notifies the *majordomo* that it has successfully occupied a locker. If the *guardian agent* decides not to accept the incoming agent into the locker, it notifies to the *locker rent agent* that is has to leave, if it does not leave immediately the *guardian agent* expels it. If the *locker rent agent* is rejected from the place because there is no space available for it, then it can travels to the next place in its *itinerary*. If the *locker rent agent* is rejected because the user is not allowed to use the service, then the *locker rent agent* would not be allowed in any *Locker Place* of the system. If the *locker rent agent* returns unsuccessfully to the *majordomo*, the *ma-*

User:	Group:	Locker Rent Agent:	Locker Agent:	Locker Content:
Id. User	Id. Group	Id. Locker Rent	Id. Locker	Id. entry
Access Key	Access Key	Creation Date	Creation Date	Date
Access Level	Access Level	Assigned Directory	Directory assigned	Origin
Description	Description	Locker Type(pri/shared)	Total Max. space	Path
Id. Locker Rent	Creation Date	Id. User	Space requested	Size
Id. Group	Max. No. Members	Id. Locker	Actual space occupied	Private(y/n)
	Id. Locker	Move Automatically(y/n)	Access key	Description
		Arrival Date	Locker Type(pri/shared)	Type(e-mail, SQLresults...)
			Max. No. Users	Priority
			Access Level	Id. Locker Rent
			Id. group	Id. Locker

Figure 4. Basic information stored in the *Locker Place* DB.

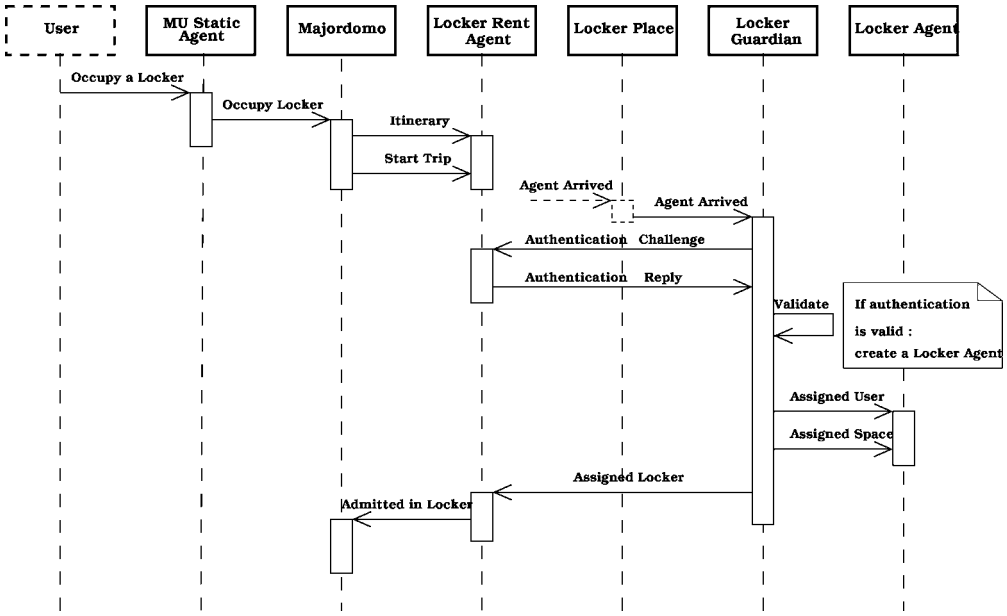

Figure 5. Interaction diagram for a successful attempt of occupying a private locker.

jordomo may decide either to try again later and/or notify the situation to the user.

4. Features of the Locker Service

In this section, we present the dynamic management of the storage space, and some aspects related to security.

4.1. Dynamic management of storage

Lockers are implemented by a pair of agents collaborating together: the *locker rent agent* and the *locker agent*. The latter agent constitutes the locker itself. This solution allows us to manage the locker in a dynamic and flexible manner. Lockers are created as they are rented by new users. This means there are no physically separated compartments to reserve, instead *locker agents* are created with space assigned in which they can store data.

The *locker guardian agent* creates a directory for each *locker agent*, and provides the *locker agent* with the path and the proper access rights so that it can create, read and write files in this directory. The *locker guardian agent* ensures that no *locker agent* access any directory other that its own, and monitors the total disk space the *locker agent* directory is using. No other agent has access to this storage space. The operating system behind the agent system and its file system manage the actual physical storage space and the related aspects. With this solution, each *locker agent* occupies disk space according to its needs and limitations. When it needs more space, the agent occupies it, and when it needs less, the agent releases space. The space used by the locker is exactly the storage used by the data saved there. Although there is a limit in how much space the agent can use (by contract or specification), flexible policies can be used to optimize the use of space and improve the service provided.

Information is saved in the DB of the *Locker Place* for each data entry stored in the lockers, about its origin, owner, location, and locker. This information allows the system and the *locker guardian* agent to monitor the use of resources and the accounting, and also to easily localize all the entries of a specific locker or user.

4.2. Security

In the context of mobile agents, security is always a principal concern because generally agents can not trust the hosts they visit, and hosts can not trust the agents they receive. The protection of agents against malicious hosts is an open problem still without a completely satisfactory solution. The approach most commonly used is to rely on an organizational framework whose hosts can be trusted [21]. This is the case in our system, since we assume that the services offered by the GSN are provided within a trusted cellular phone company. Similarly, users are not unknown to the system, since they have signed a contract with the company for renting the services, and they have been assigned an ID and a set of passwords. Thus, the system can authenticate and validate authorized users and their agents.

The security of hosts against agents can be achieved satisfactorily by using existing techniques [21]. The central issue is to prevent uncontrolled access. In our system, the only agents that come from the outside, i.e., are created outside the GSNs, are the *majordomos*. However, the *majordomo* agents classes must have been programmed, released and signed by a known entity. It must be ensured that such agents represent authorized users. To achieve this, credentials are associated with the agents. Credentials correspond to the identity of their user, access level and key. Such credentials are used not only for the authentication and authorization of the majordomo agents, but also for controlling their access to

services and data. This is achieved by checking their credentials against the information the system maintains about the registered users. When an agent creates another agent, it passes part of its identification information to the new agent by issuing it a "passport" containing the necessary information to identify the agent, its creator, and so, to its user. This "passport" is sensitive information that needs to be protected against copies, thieves and modifications.

Another important issue is preventing uncontrolled use of the host resources. The programming languages and the agent platforms used provide mechanisms to prevent agents and hosts from accessing data or code of others such as separated memory address spaces for the host and each agent. Also, they provide authentication and authorization mechanisms, as well as mechanisms to avoid uncontrolled copying and cloning of agents. To authenticate code and its origins, cryptographic techniques can be used, as for example signatures [20].

Apart from the use of mobile agents, in the *Locker Rental Service*, data stored in a private locker belong to the user and are protected against unauthorized access from other users or their agents. In a shared locker, members of the group owning the locker are allowed to access the data that is specifically stored as common, while each user in the group has a private area where he can store information that nobody else has the right to access. This is achieved by not allowing the *locker agents* access any other disk space but their own. When a *locker rent agent* makes a request to a *locker agent*, the *locker agent* checks the authenticity of the *locker rent agent* and decides to which data it has access to.

Finally, the use of the resources of the *Locker Place* must be controlled by the system in order to monitor the use of space, administrate the resources and avoid intrusions, as well as to implement the accounting mechanisms and policies.

5. Mobility of the lockers

In ANTARCTICA, the *majordomo* agent that belongs to a mobile user is located at the GSN under whose coverage the MU is situated. This means that when the MU moves to an area covered by a different GSN, its *majordomo* agent also moves to the new GSN, carrying with it its data and some of the agents that are working for it. The question we address in this section is whether to also move the locker. When deciding to occupy a locker, the user specifies whether he wants: (a) the locker to always move following his movements, (b) the locker to remain stationary, (c) the system to decide when to move the locker, or (d) to combine the use of the locker with the use of a small cache at the *majordomo*.

When option (a) is chosen, i.e., the user specifies that he wants the locker to follow his movement, there is a possibility that neither the corresponding GSN nor any GSN in the neighborhood can accommodate the locker. In this case, the system informs the user that the locker can not be moved. With option (b), the locker is created at the GSN, under whose coverage the MU is placed at the time the user asks for occupying a locker, and stays there until it is released.

With option (c), the decision whether to move a locker or not is taken by the system (transparently from the user) based on the cost-benefit of the transmission. The benefit of moving the locker stems for the fact that the cost of the interactions (*delay time*) of the two agents, the locker and the *majordomo*, is reduced since the agents are located in near-by sites. In order to move the locker, first the *Locker Place* negotiates to find another *Locker Place*, located at the new GSN where the *majordomo* is situated, or close to it, which can accept the locker. Notice that in the best case, the *Locker Place* that accepts the incoming locker is the one situated at the same host that the user *majordomo* agent, and so, the communications between the locker and the *majordomo* will be local at the host, without going through the network. However, in any case, the *Locker Place* accepting the incoming locker will be at least closer, in terms of network communications response time, than the original one.

Briefly, the cost of moving the locker (C_{Move}) consists of the cost of transferring all data stored in it (C_{Transf}) plus the cost of interacting with the *majordomo* in the GSN (C_{CloseInt}). The cost of not moving the locker is the cost of the interaction between the *majordomo* and the locker in the case they are located at different GSNs (C_{NotMove}).

The cost of the interaction[4] of the *majordomo* and the locker can be described using three factors:

(1) T_{stay} (s): an estimation of the time the *majordomo* will stay in the new GSN. The *majordomo* can estimate this from the previous behavior of the user, or the user can explicitly provide such information;

(2) N_{Requests} (du/s): an estimation of the interaction rate between the *majordomo* and the locker, measured in terms of number of data units per time unit, that can also be induced from previous interactions; and

(3) C_{Request} (s/du): the cost of processing a data request between the *majordomo* and its locker measured in time units per data unit;

where C_{Request} is the term affected by the distance between the *majordomo* and its locker. The following parameters are involved in the computation of C_{Request}, and so in C_{CloseInt} and C_{NotMove}:

(1) S_{Avg} (kb/du): an estimation of the average size of the data units to be interchanged.

(2) C_{FarCm} (s/kb): the cost of communications between the *majordomo* in the new GSN and its locker in the current position. This can be calculated by the system.

(3) C_{NearCm} (s/kb): similar to the previous cost but for the case of the locker being located in a position near to the new GSN.

To compute C_{Transf}, the following must be considered:

[4] By interaction we consider only requests for data, that is, operations for which the user, or the *majordomo* agent, waits for an answer. The *majordomo* can request update operations on the locker while at the same time it continues working on other tasks.

(1) T_{CrLAg} (s): the time to create a new *locker agent* in the new *Locker Place*,

(2) T_{Send} (s/kb): the time to transmit to the new *Locker Place* a Kb of data stored in the locker,

(3) S (kb): the total size of the data stored in the locker,

(4) T_{MoveLRAg} (s): the time to move the *locker rent agent* from the old *Locker Place* to the new one.

Without going into details and without considering the cost of interchanging few messages among the agents in order to coordinate the movement, as well as the deletion of what remains of the old locker, the two costs under consideration are detailed in equations (1) and (2). The locker is transferred if C_{NotMove} is greater than C_{Move}, or equivalently, when the ratio C_{Move} / C_{NotMove} is smaller than 1. Equation (3) shows the calculation of this ratio:

$$\begin{aligned} C_{\text{NotMove}} &= T_{\text{stay}} \times N_{\text{Requests}} \times C_{\text{Request}}, \\ &\text{where } C_{\text{Request}} = C_{\text{FarCm}} \times S_{\text{Avg}}. \end{aligned} \tag{1}$$

$$\begin{aligned} C_{\text{Move}} &= C_{\text{CloseInt}} + C_{\text{Transf}}, \\ &\text{where } C_{\text{CloseInt}} = T_{\text{stay}} \times N_{\text{Requests}} \times C_{\text{NearCm}} \times S_{\text{Avg}}, \\ &\quad C_{\text{Transf}} = T_{\text{CrLAg}} + S \times T_{\text{Send}} + T_{\text{MoveLRAg}}. \end{aligned} \tag{2}$$

$$\begin{aligned} \frac{C_{\text{Move}}}{C_{\text{NotMove}}} &= \frac{C_{\text{CloseInt}} + C_{\text{Transf}}}{C_{\text{NotMove}}} \\ &= \frac{C_{\text{CloseInt}}}{C_{\text{NotMove}}} + \frac{C_{\text{Transf}}}{C_{\text{NotMove}}} \\ &= \frac{C_{\text{NearCm}}}{C_{\text{FarCm}}} + \frac{C_{\text{Transf}}}{C_{\text{NotMove}}}. \end{aligned} \tag{3}$$

In general, moving the locker from one GSN to another is an expensive operation. However, notice that the locker is not moved at the time the user makes the first request from his new location, but after the *majordomo* itself moves to the new GSN. This means that the movement of the locker can be performed asynchronously without directly delaying the user's or the *majordomo*'s operation. Furthermore, the possibility of moving a locker is useful not only because this way the data are closer to the user, but also because the system can decide to move a number of lockers from one GSN to another to balance the system load, while taking care of not deteriorating the interaction between the lockers and their *majordomos* by overly incrementing the cost of their communications.

Option (d), i.e., to combine the use of the locker with the use of a small cache by the *majordomo*, is based on the fact that all data obtained by the agents, or sent to/by the user, actually passes through the *majordomo*. This way, the *majordomo* can maintain a small cache feeding it with the data most commonly used by the agents and the user. We refer to this piece of data as "hot-data". This cache is built and feeded with data while the *majordomo* continues serving requests from the user. Of course, this cache must be kept small since the *majordomo* cannot keep with it large amounts of data. The management of a cache by the *majordomo* introduces an overhead in its operation since not only has it to access the cache, but also handle cache replacements (for example, applying LRU) and maintain cache consistency. However, in general this extra cost will be compensated by the benefit obtained because the *majordomo* will be able to quickly serve the user and other agents requests over these data.

Using this approach, the cache will always move with the *majordomo*, and so, the size of the cache does not has influence in the decision of moving or not the locker, although the formulas to determine whether the locker must be moved or not are now different[5].

Briefly described and not considering the small cost of checking the cache whenever a data is being requested, the cost of processing a request for data (C_{Request} (s/du)), is the cost of retrieving this data from the cache (C_{AccCache}) or the cost of retrieving it from the locker ($C_{\text{AccLocker}}$), and α is the probability of finding the data in the *majordomo* cache, i.e., the cache hits.

Therefore, equations (1) and (2) to decide when the locker must be moved or not are modified. There is a new parameter involved in the computation of the formulas: T_{AccCache} (s/kb), which is the time needed to retrieve an item from the cache. Therefore, equations (1) and (2) are now rewritten as shown in equations (4) and (5), respectively, while the ratio formula is now equation (6):

$$\begin{aligned} C_{\text{NotMove}} &= T_{\text{stay}} \times N_{\text{Requests}} \times C_{\text{Request}}, \\ &\text{where } C_{\text{Request}} = \alpha \times C_{\text{AccCache}} + (1-\alpha) \times C_{\text{AccLocker}}, \\ &\quad C_{\text{AccCache}} = T_{\text{AccCache}} \times S_{\text{Avg}}, \\ &\quad C_{\text{AccLocker}} = C_{\text{FarCm}} \times S_{\text{Avg}}. \end{aligned} \tag{4}$$

$$\begin{aligned} C_{\text{Move}} &= C_{\text{CloseInt}} + C_{\text{Transf}}, \\ &\text{where } C_{\text{Transf}} = T_{\text{CrLAg}} + S \times T_{\text{Send}} + T_{\text{MoveLRAg}}, \\ &\quad C_{\text{CloseInt}} = T_{\text{stay}} \times N_{\text{Requests}} \times C_{\text{Request}}, \\ &\quad C_{\text{Request}} = \alpha \times C_{\text{AccCache}} + (1-\alpha) \times C_{\text{AccLocker}}, \\ &\quad C_{\text{AccCache}} = T_{\text{AccCache}} \times S_{\text{Avg}}, \\ &\quad C_{\text{AccLocker}} = C_{\text{NearCm}} \times S_{\text{Avg}}. \end{aligned} \tag{5}$$

$$\begin{aligned} \frac{C_{\text{Move}}}{C_{\text{NotMove}}} &= \frac{C_{\text{CloseInt}} + C_{\text{Transf}}}{C_{\text{NotMove}}} \\ &= \frac{C_{\text{CloseInt}}}{C_{\text{NotMove}}} + \frac{C_{\text{Transf}}}{C_{\text{NotMove}}} \\ &= \frac{\alpha \times T_{\text{AccCache}} + (1-\alpha) \times C_{\text{NearCm}}}{\alpha \times T_{\text{AccCache}} + (1-\alpha) \times C_{\text{FarCm}}} \\ &\quad + \frac{C_{\text{Transf}}}{C_{\text{NotMove}}}. \end{aligned} \tag{6}$$

Finally, the shared lockers are associated with a group of users whose movement is in general independent. Their location is the result of a decision of the group or of the representative of the group, so shared lockers do not necessarily move following their users.

[5] Notice that the *majordomo* stores in its cache the data most commonly accessed, this means it may stores data that is not also stored in the locker. In order to simplify the formulas we abstract this fact and consider only requests for data also stored in the locker.

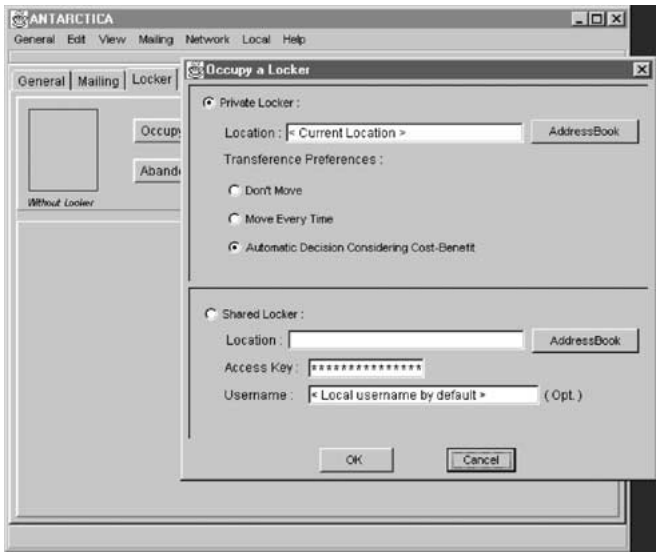

Figure 6. Screenshot of the *Locker Rental Service* GUI displayed at the MU.

6. Implementation and performance results

Our prototype is being implemented using the "Aglets Workbench" [9] platform. Currently, it supports the creation of private lockers, the storage of files and sql results in the lockers, authorized access to the data stored, the movement of the locker and the release of the lockers. Also, we have fully implemented the interaction among all components involved in the service: the agents (*MU static agent*, *majordomo*, *locker guardian agent*, *locker rent agent* and *locker agent*), the places (MU, Inventory and Locker) and the database in the *Locker Place*.

In our experiments, we assume that the MU has enough capacity to support a graphical interface (see figure 6). The experiments have been performed using on the one hand, a portable computer equipped with a 233 MHz Pentium processor, 64 M of RAM, a 9600 bps GSM card as the communication device, and running Windows 98. This machine plays the role of the MU. On the other hand, we used a fixed computer equipped with a 400 MHz Pentium II with 128 M of RAM running Linux Red Hat 6.1. This machine plays the role of the GSN.

6.1. Basic performance results

We present in this section two preliminary experiments. The first one measures the time for the creation of a locker. The second one compares the performance obtained accessing remote files from a mobile device using wireless communications and three different approaches[6].

6.1.1. Locker creation

The goal of the first experiment is to measure the average time it takes to create a private locker. In order to initiate the creation process, the MU sends a message to its majordomo requesting the occupancy of a locker. The time it takes to send that message from the MU to the GSN is affected by the communication media used. For wireless communications, using a 9600 bps GSM card, we measured 2335.5 ms on the average with a standard deviation of 39.26 ms. After sending this message, the MU can be disconnected since the *majordomo* takes the responsibility for the negotiations for the creation of the locker. The time it takes for the *majordomo* to create the *locker rent agent*, send it and receive confirmation of the occupancy is 253 ms on the average, with a standard deviation of 213 ms.

6.1.2. File fetching

We performed several experiments, in which we compared the time that the MU needs to be connected in order to get, for example, a file or a web page, according to the following three different approaches:

(a) Client/Server (CS): the MU opens a connection directly to the address where the file is located and the connection remains open until the file is downloaded.

(b) Client/Agent/Server (CAS): the MU sends a request message to its *majordomo* agent located in the GSN, specifying the address of the file to be downloaded. Then, the MU closes the connection. The *majordomo* obtains the file and when the MU connects again, it sends the file to the MU in one answer message. With this approach, the minimum time the MU needs to be connected to get a file is just the time needed to send and receive the request and answer messages.

(c) Client/Agent/Server approach combined with the use of a locker (CAL): the MU sends a request message to its *majordomo* agent located in the GSN to obtain a file and store it in the locker. Then, the MU closes the connection. The *majordomo* agent obtains the file and sends it to the locker. Once connected again, the MU sends a message (read message) to the *majordomo* to read the file from the locker. Then, the *majordomo* sends the file in one answer message. With this approach, the time the MU needs to be connected to get a file is the time needed to send the request, read and answer messages. Notice that in this approach we are assuming the worst case. If the majordomo detects that the MU is connected again when having the file stored in the locker, it may decide to send the file to the MU, without waiting for a specific request from the user. In this case, the time associated with the read message is eliminated.

When comparing the CAS with the CAL approach, the use of the locker in the CAL approach introduces a time overhead due to the read message, but the ability of storing data for the MU in a persistent and safe way constitutes the strong point of the CAL approach over the CAS one. Notice that we are only interested in the MU connection time and not in the total cost. Saving connection time also means that the MU saves battery power and money. The CAS and CAL approaches make it possible to work in an asynchronous way, i.e., the MU submits a task and then disconnects or continues working on

[6] Each experiment has been performed more than 20 times.

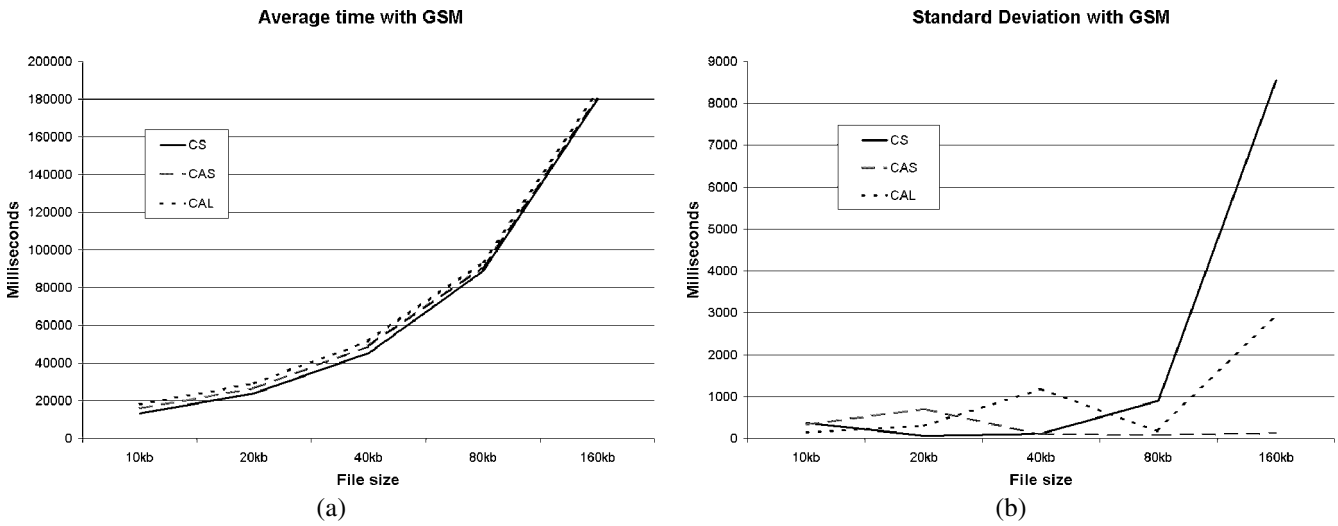

Figure 7. Using GSM. (a) Comparison of the minimum time the MU has to be connected in order to get a file by each of the approaches. (b) Standard deviation of the times in (a).

Name	File Type	Original Size	Compressed Size	Compression percentage	Average time for compression
test2.pdf	text	161.76 Kb	154.72 Kb	4	146 ms
test1.pdf	text	184.96 Kb	154.55 Kb	16	180 ms
test1.au	audio	172.14 Kb	90.02 Kb	48	237 ms
test1.mid	audio	153.53 Kb	8.96 Kb	94	45 ms

Figure 8. Characteristics of the set of files used for the compression tests.

other tasks, while the submitted task is being executed in the GSN. When the task finishes, the MU receives the results.

For this experiment we used a set of 5 files of different sizes (10 Kb, 20 Kb, 40 Kb, 80 Kb and 160 Kb) placed in a host situated 9 hops away from the MU and 9 hops away from the GSN. Each file was downloaded to the MU more than 20 times for each of the approaches using the wireless communication media (GSM).

Figure 7 shows the results obtained when using wireless communications. The CS approach performs slightly better than the approaches using agents. The time the MU needs to be connected is determined by the time it takes to interchange data between the MU and the GSN. With the CS approach that time is mostly the time it takes to send the file through the wireless media. However, in the CAS and CAL approaches there are some messages interchanged between the agents situated in the MU and in the GSN. Even if these messages are short, the transmission of them through the GSM media at 9600 bps causes the small overhead of those approaches over the CS one. However, the larger the file, the more irrelevant the overhead.

The analysis of the standard deviation (figure 7(b)) shows how with the CS large file sizes result in long connection times and unstable behavior. The CAS and CAL standard deviation is lower.

6.2. Improving the performance

Previous results do not constitute a proof against the use of the CAS and CAL approaches. On the contrary, taking into account that the cost that influences the most the final cost is the one associated with the low speed communication media, the approaches using the intermediary agents give the opportunity to reduce the communication cost, for example by: (a) applying filters in the GSN that reduce the amount of data to be transmitted, (b) avoiding interactions between the MU and the GSN, and (c) performing tasks in the GSN that otherwise should be executed in the MU (with the higher use of low speed communication media that this implies).

6.2.1. Applying filters

In this experiment, a compression filter was applied to the files before transmitting them to the MU. The filter applies a *lossless compression*, that means that there is no loss of information. Note that we could have obtain even better results, if a more sophisticated compression algorithm was used. This would have been possible since the *majordomo* agent has information about the characteristics of the MU and the preferences of the user. For this test, we used a set of sample files with sizes between 153 Kb and 185 Kb, containing different types of data and presenting different compression ratios. Figure 8 shows the characteristics of the files we used. Figure 9(a) shows the mean of the time the MU has to be connected to get the files by each of the approaches considered, again using the wireless media. Figure 9(b) shows the corresponding standard deviation. These results show how even a small compression percentage is enough to make the approaches with agents outperform the CS approach.

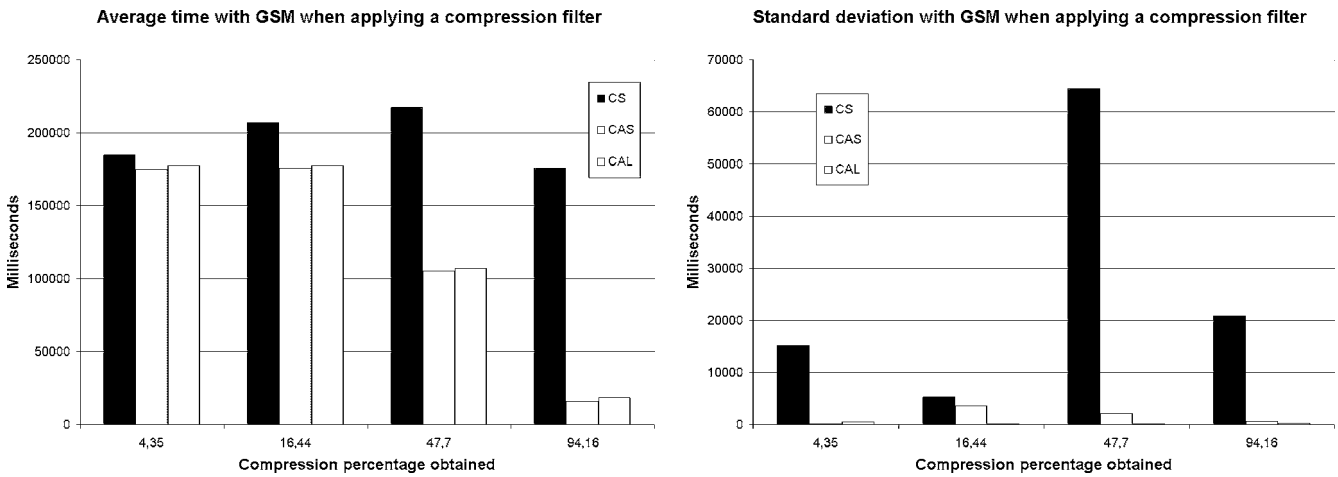

Figure 9. Using GSM. (a) Comparison of the minimum time the MU has to be connected in order to get a file by each of the approaches. (b) Standard deviation of the times in (a).

6.2.2. Performing tasks in the GSN

In the previous experiment, we can observe that the performance of the CAL approach is slightly worse than the behavior of the CAS approach. This is because the use of the locker in the CAL approach introduces an overhead due to the extra message sent from the MU to the *majordomo* in the GSN, once the MU reconnects. We next present an experiment that shows how the small overhead of the CAL approach is compensated and even inverted, by using a feature provided by the locker to facilitate dealing with complex requests.

For this experiment, we assume that the user requests, in a single request message, 4 different web pages to be obtained and stored in the locker. Later on, he requests (one read message) to download them together to the MU (one answer message). We compare this scenario with the following one in which the user requests each page, one by one (one request and one answer message per page), to be obtained and downloaded to the MU. Notice that this last scenario corresponds to the case in which the user has not contracted a locker. Therefore, the *majordomo* agent can only keep a limited amount of data with it for a short period of time before sending them to the MU.

Figure 10 shows how avoiding messages between the MU and the GSN makes the CAL approach better than the CAS one. The total amount of data transmitted, belonging to the web pages, is the same in both cases, the only difference between both approaches is the number of messages interchanged. The analysis of the standard deviation also shows how the larger the number of messages interchanged the larger the instability.

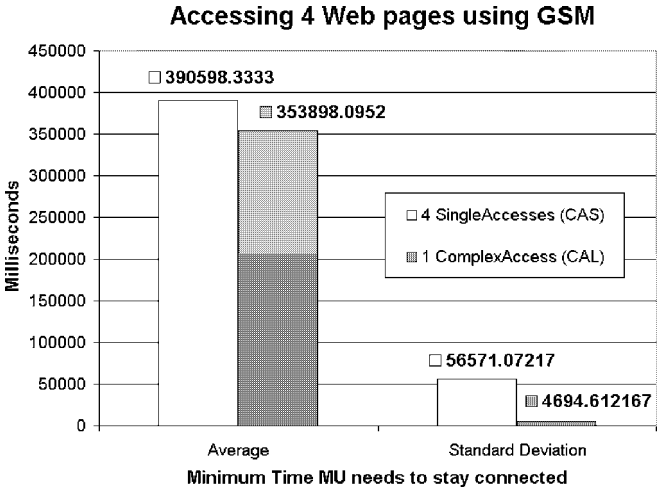

Figure 10. Using GSM. Comparison of the minimum time the MU has to be connected in order to get 4 web pages using the CAS and the CAL approaches.

7. Related work

There are two lines of research related to our work. The first line includes research that relies on the idea of renting resources: for example, in the area of *Metacomputing*, the Java Market project [1] allows any host to submit (Java) processes to be executed in any host in the Internet that has available resources and is willing to rent its computational capacity. However, in this work the execution of processes is considered only in terms of CPU cycles and communications consumption, while the main idea of the locker is not the execution of processes but the rental and occupancy of "storage" that processes working for the user can use to maintain data. Furthermore, in the Java Market, any host can participate by sending or receiving processes, while in our system, the GSNs that provide the locker space, are servers dedicated to offer system services. The GSN acts as a proxy or intermediary element situated close to the MU. The second line of related research exploits the use of *proxies* and/or software agents [8,11,13,17]. To our knowledge, our work is the first one to combine both aspects, by employing proxies and agents to provide users with storage external to their mobile computers. Finally, our work can benefit from other works, such as the CODA project [12], by implementing the ideas

and techniques they propose for the maintenance of data consistency among copies stored in the MU cache, the GSN and the servers.

Several commercial products are appearing, such as X-Drive [22], Driveway [3], mySpace [23], that offer storage space in the Internet. Their fast popularization, the number of products appearing, and the number of customers, demonstrate the interest in this kind of services. Using these products, users must keep a connection open for the whole time they are using this space. If the connection breaks, they must log on again. More expensive commercial products also appear, called digital vaults or virtual safe-deposit boxes, where users can safely deposit their legal, vital or important documents. Their aim is to provide the same service that bank safe deposit boxes offer to their customers, and so they assure the privacy and safety of the stored documents against theft or disaster. Some of these products are: FileTrust [5], DigiVault [14], Zephra [24] and PrivateArk [2].

When compare to our proposal, such products constitute a plain service: a passive storage space where to store and from where to download files. They require the direct intervention of the user, and the rented space is situated always at the same location, that is, in the safe hosts of the company. ANTARCTICA *Lockers Rental service* goes a step further:

(1) the locker can follow the user in his movements, so it stays closer to the user's actual physical location;

(2) due to the use of the agent technology, data in the lockers can be used by agents working on behalf of the user, representing him in the network even while the user is actually disconnected. These agents take care of optimizing the data format and their transmission process, dynamically considering the communications resources available, the user preferences and the specific mobile device resources and characteristics; and

(3) the *Locker Rental Service* provides support to other services offered by ANTARCTICA, which actually feed it with data and increment the functionalities and benefits the user can get from it.

8. Conclusions and future work

In this paper, we present a new data service for the mobile users: the *Locker Rental Service*. This service allows its users to use ubiquitous persistent storage space located at the fixed network, outside of their mobile computer but close to them, accessible anywhere anytime. Besides providing an extension of the storage space of the mobile computer, the service gives to its users more autonomy. Autonomy means that users can decide when to work disconnected or connected to the wireless network, since they can always count on the locker space for storing both the messages sent to them by other users and the results of their requests. Moreover, the use of lockers allows reducing data traffic in the wireless network, and provides an alternative for storing sensitive data. For the implementation of the *Locker Rental Service* we use the agent technology. The induced overhead of using agents is greatly compensated by the advantages they provide with respect to performance, flexibility and adaptability. Our experiments have shown the feasibility of the implementation when compared to the traditional Client/Server approach. Moreover, we have shown how its benefits can be improved when complex tasks are considered.

Our plans for future work include experimentation with the locker mobility feature and evaluation of its performance with and without cache; as well as a performance comparison with other related commercial products.

Acknowledgements

This work is supported by CICYT: Comisión Interministerial de Ciencia y Tecnología, Spain [TIC2001-0660], and MOVISTAR, a Spanish Cellular Phone Company.

References

[1] Y. Amir, B. Awerbuch and R.S. Borgstrom, A cost-benefit framework for online management of a metacomputing system, The International Journal for Decision Support Systems 28(1–2) (April 2000) 155–164.
[2] Cyber-Ark Software Ltd., `http://www.cyber-ark.com/`
[3] Driveway Corporation, `http://www.driveway.com/`
[4] M. Esler, J. Hightower, T. Anderson and G. Borriello, Next century challenges: Data-centric networking for invisible computing: The Portolano project at the University of Washington, in: *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCOM'99)*, Seattle, WA (August 1999).
[5] Fleet Boston Financial Corporation, `http://www.filetrust.com/`
[6] General Magic, Mobile agents white paper, in: *General Magic Technology White Paper* (1998).
[7] A. Goñi, A. Illarramendi, E. Mena, Y. Villate and J. Rodríguez, Antarctica: A multiagent system for internet data services in a wireless computing framework, in: *NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, Scottsdale, AZ (October 2001).
[8] R. Gray, D. Rus and D. Kotz, Agent TCL: Targeting the needs of mobile computers, IEEE Internet Computing (1997).
[9] IBM Aglets Workbench Home Page, `http://www.trl.ibm.co.jp/aglets/`
[10] T. Imielinski and B.R. Badrinath, Mobile wireless computing: Challenges in data management, Communications of the ACM (October 1994) 19–27.
[11] A. Joshi, S. Weerawarana and E. Houstis, On disconnected browsing of distributed information, in: *IEEE RIDE97* (1997) pp. 101–107.
[12] J.J. Kistler and M. Satyanarayanan, Disconnected operation in the Coda file system, ACM Transactions on Computer Systems 10 (1992) 3–25.
[13] E. Kovacs, K. Röhrle and M. Reich, Mobile agents OnTheMove – integrating an agent system into the mobile middleware, in: *Acts Mobile Summit*, Rhodos, Greece (June 1998).
[14] Lexias Inc, `http://www.mydigivault.com/`
[15] D. Milojicic, Mobile agent applications, IEEE Concurrency 7(3) (1999) 80–90.
[16] Milojicic et al., MASIF, The OMG Mobile Agent System Interoperability Facility, in: *Proceedings of Mobile Agents'98* (September 1998).
[17] S. Papastavrou, G. Samaras and E. Pitoura, Mobile agents for WWW distributed database access, in: *Proceedings of the International Conference on Data Engineering* (1999).
[18] E. Pitoura and G. Samaras, *Data Management for Mobile Computing* (Kluwer Academic, 1998).

[19] G. Samaras and A. Pitsillides, Client/Intercept: a computational model for wireless environments, in: *Proceedings of the 4th International Conference on Telecommunications (ICT'97)* (April 1997).
[20] B. Schneier, *Applied Cryptography*, 2nd ed. (Wiley, 1996).
[21] C. Tschudin, Mobile agent security, in: *Intelligent Information Agents – Agent Based Information Discovery and Management in the Internet*, ed. M. Klusch (Springer-Verlag, 1999) pp. 431–446.
[22] Xdrive Technologies, `http://www.xdrive.com/`
[23] YourZ.com Inc., `http://www.myspace.com/`
[24] Zephra Corporation, `http://www.securitymadesimple.com/`

Yolanda Villate received her BSc from the Computer Science Faculty of the University of the Basque Country, Spain, in 1997. Next, for three years she was a research assistant at the University of the Basque Country. During 1999 she was a visiting scholar at the Computer Science Department at Purdue University, USA. Since February 2001, she is an Assistant Professor at the Computer Science and Systems Engineering Department, University of Zaragoza, Spain. Her main research interests, and PhD work, are focused on data management for mobile computing. Her publications include several conference and workshop articles on the subject of mobile computing and data management.
E-mail: yvillate@posta.unizar.es

Arantza Illarramendi received her BSc from the Computer Science Faculty of the Technical University of Madrid, Spain, and her PhD in computer science from the Basque Country, Spain. Since January 1995, she is a Full Professor at the Computer Science Faculty of the University of the Basque Country. Her main research interests are data management for mobile computing and query processing for global information systems. Her publications include several journal and conference articles and a recently published book on ontology-based query processing for global information systems. She is a member of the IFIP WG 2.6 and has served on a number of program committees.
E-mail: jipileca@si.ehu.es

Evaggelia Pitoura received her BSc from the Department of Computer Science and Engineering of the University of Patras, Greece, in 1990 and her MSc and PhD in computer science from Purdue University in 1993 and 1995, respectively. Since September 1995, she is on the faculty of the Department of Computer Science of the University of Ioannina, Greece. Her main research interests are data management for mobile computing and multidatabases. Her publications include several journal and conference articles and a recently published book on mobile computing. She received the best paper award in the IEEE ICDE 1999 for her work on mobile agents. She has served on a number of program committees and was a program co-chair of the MobiDE01 and MobiDE99 workshops held in conjunction with Sigmod'01 and MobiCom'99, respectively.
E-mail: pitoura@cs.uoi.gr