Agent-Based and Mobile, External Storage for Users of Mobile Devices *

Yolanda Villate
CSSI Depart.
Univ. of Zaragoza
Maria de Luna 3, 50018
Zaragoza, Spain
yvillate@posta.unizar.es

Arantza Illarramendi
CLS Depart.
Univ. of the Basque Country
Apdo. 649, 20080
San Sebastian, Spain
jipileca@si.ehu.es

Evaggelia Pitoura
CS Depart.
Univ. of Ioannina
GR-45110 Ioannina,
Greece
pitoura@cs.uoi.gr

ABSTRACT

Keeping data in a secure and safe space is a real problem for many users of mobile devices because those devices are more vulnerable and fragile than the stationary ones. To solve that problem many commercial products are appearing that offer storage space in the Internet. In this paper, we present a new solution to the mentioned problem, based on the use of the agent technology, called the *Locker Rental Service* that goes a step further with respect to existing ones. Our service provides two specific advantages: autonomy and mobility. In terms of *autonomy*, data in the lockers are managed by agents working on behalf of the users, representing them in the network even while the users are actually disconnected. These agents take care of obtaining requested data, optimizing their format and their transmission, taking into account the communication resources available, the user preferences and the specific mobile device resources and characteristics. With respect to *mobility*, lockers can follow the users in their movements to remain close to the users' actual physical location.

Keywords

Mobile computing, multi-agents systems, data storage, mobile services.

1. INTRODUCTION

Nowadays we are witnessing a proliferation of mobile devices that give users the possibility of accessing and/or receiving local or remote data anywhere and at anytime. However, when working with these kind of devices, users face many obstacles related with the nature of the devices and the communication media used. Even if future devices may not be as resource-limited, they will still be constrained in terms of size, power consumption, and intermittent connectivity. Furthermore, by disconnecting during idle periods, they will consume less power.

To overcome 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 utilize an agent-based framework in which agents operate autonomously on the user behalf. In particular, we are developing the ANTARCTICA[1] system [6] to provide a set of data services that enhance the capabilities of mobile devices and offer new possibilities to mobile users. The architecture of ANTARCTICA is based on two pillars: the software agents technology and the Client/Intercept/Server (CIS) model [17, 16] and so it incorporates modules and agents both at the mobile devices[2] and at intermediary elements, or proxies, situated in the fixed network called the *Gateway Support Nodes*[3](GSNs). In this paper we focus on a service central to ANTARCTICA: the *Locker Rental Service*. This service incorporates mechanisms that allow mobile users to rent storage space, called **lockers**, at GSNs.

The *Locker Rental Service* offers the following advantages to its users: **Storage space:** The locker becomes an extension of the mobile device's disk placed at the fixed network. Thus, a mobile client can deposit in the locker all required data and results obtained (within the capacity limitation established). **Data protection:** The mobile client is the owner of the data stored in the locker. 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 mobile device, such as configuration files or files containing sensitive information, that need to be protected against losses. Besides those advantages, that as we show in the following section are also provided by other systems, the use of the

*This work is supported by *CICYT: Comisión Interministerial de Ciencia y Tecnología*, Spain. [TIC2001-0660]; DGA project P084/2001; and MOVISTAR, a Spanish Cellular Phone Company.

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

[2]In this paper the terms mobile device and mobile unit are considered synonymous.

[3]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.

agent technology for the implementation of the service, provides the following specific advantages: **Higher presence at a lower cost:** software agents manage autonomously the locker contracted by the mobile client, so the user can stay disconnected for longer periods of time. The agents can retrieve and manipulate data stored in the locker and they can also store new data in the locker. **Wireless communications optimizations:** The space in the locker can be used to store data until the agent that manages the locker considers it possible or desirable to send them to the mobile device. Before sending data, the agent can preprocess them, filter or adapt them to the needs of the mobile device and its user. **Flexibility and adaptability:** The implementation of the 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. **Proximity:** agents physically supports mobility of the lockers to follow the movements of the mobile device and be always close[4] to its current location.

Two basic kinds of lockers are provided by this service: private and shared. A **private locker** is related to a single user; the data stored belongs to that particular user and can be accessed or modified only by authorized agents representing the user. 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. Moreover, the space in the locker that is shared by all users of the group constitutes an encounter place for these users who are able to communicate, share data, interchange messages, write messages for the rest of the group in a blackboard service provided and collaborate together. A user may have one private and several shared lockers. In this case, the user must choose one of the lockers as the principal one in which to maintain e-mails, mobile device sensitive files or backup copies.

The ANTARCTICA *Locker Rental Service* and some of its features were introduced previously in [20], [22] and [21], where some experimental results of the service were presented and the lockers mobility problem was first discussed. In this paper, we develop further the mobility problem and show different alternatives of it as well as some experimental results obtained.

The rest of this paper is structured as follows. First, we compare our system with related work. In Section 3, we present a general overview of the *Locker Rental Service*, and elaborate on the different agents involved in it, along with some security considerations. In Section 4, we study the mobility of the lockers. In Sections 5 and 6, we present advanced features of the mobility and show performance results. Finally, we offer our conclusions.

[4]In this paper we mean close in term of network communications response time.

2. RELATED WORK

We consider approaches related to ours from different perspectives: first we compare our work with other research works that also offer storage space, then we compare with research related to proxies and agents, and finally with emerging commercial products that also offer space to rent.

In the first line of related research we include the OceanStore project[12] and the Chord project[2]. The OceanStore project relies on the idea of renting storage space. This project builds a global persistent data storage architecture that seeks to transparently improve clients data access performance by supporting *nomadic data*, that is, using promiscuous caching of clients data in the OceanStore servers. But this work does not provide autonomy, and mobility is got by replicating data which implies the need of more space and costly update operations. The Chord project supports a cooperative storage system in which any host can voluntarily join to the community and contribute with storage space to be used to store any data published by anyone in the system. Data handed over to the system is transparently replicated and cached on different hosts. OceanStore and Chord are projects on peer-to-peer computing, where hosts are not proprietary as the GSNs in our approach.

The second line of related research exploits the use of *proxies* and/or software agents [15, 11, 7, 9]. 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 devices. However, our work can benefit from other works, such as the CODA project [10], by implementing the ideas and techniques they propose for the maintenance of data consistency among copies stored in the mobile device cache, the proxy and data servers.

On the other hand, several commercial products are appearing, such as X-Drive[23], Driveway[3], mySpace[24], that also 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. Such interest is shown from users with mobile behaviors, such as those equipped with mobile devices, mobile workers, travelers, or users with different Internet connections at work and at home, as well as those with slow connections to the network (wireless, or via modem). Many of these commercial products offers a version of its product that enable users to access their accounts from any web enabled Palm Pilot, PDA or even web-enabled cell phones. The more popular ones usually allow users to share folders with other users specifying read only and write access rights, and integrate e-mail accounts so, the user can store the attachments in its account, and can attach stored files to new mails. 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[4], DigiVault[13], Zephra[25] and PrivateArk[1].

When compared to our proposal, these commercial 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. Our *Locker 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. Notice that ANTARCTICA servers, the GSNs, host not only users data, but also processes and agents running for the users.

3. LOCKER RENTAL SERVICE. GENERAL OVERVIEW

As mentioned before, the *Locker Rental Service* is offered within the ANTARCTICA system which makes use of the software agents technology [5] and the Client/Intercept/Server (CIS) model[17, 16]. So, the *Locker Rental Service* (see Figure 1) incorporates on the one hand, agents in the mobile devices and on the other hand, agents and lockers in the intermediary elements or proxies situated in the fixed network (those proxies will be managed by the private company supporting ANTARCTICA). Places, contexts where agents can execute, are also included in the figure. In the proxies, there is a place called *Inventory* where agents are executed, and where they can get the information they need about other GSNs, places and services in the system. Besides that, there are specialist places, as for example the *Locker Place*, which is the one related to the *Locker Rental Service*. The *Locker Place* maintains a database with information about the agents populating it, the users they represent and the data kept in the lockers. In that way it controls the population and contents of the lockers in a persistent way, and can automate the process of looking for inconsistencies or fraudulent uses as well as abuses.

The software agents involved in the *Locker Rental Service* work autonomously. Moreover, the existence of the different agents and places is transparent to the user. The agents working for a user could survey the user behavior and manage a user profile, not only with his preferences, but also applying adaptive learning techniques, so they could foresee user future behavior and autonomously take decisions. However, these features are not yet implemented in the current ANTARCTICA system prototype.

3.1 Agents Involved in the Service

In general, an *agent* is considered to be a `computer program that acts autonomously on behalf of a person or organization.` It is commonly assumed that an agent is at least[5]: *autonomous*, the agent exercises control over its own actions; *reactive (sensing and acting)*, the agent responds to changes in the environment; *goal-oriented*, it does not simply act in response to the environment but its behavior is directed towards a goal; *temporally continuous*, the agent is a continuously running process. Agents can also have other optional features that can be used to classify them. For example, an agent can be mobile, that is, able to transport itself from one machine to another.

[5]It follows the agent terminology of the *Mobile Agent System Interoperability Facilities Specification* (MASIF)[14].

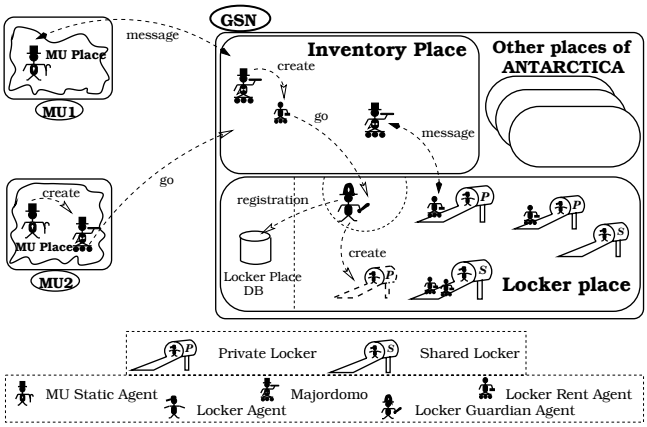

Figure 1: Elements involved in the *Locker Rental Service*.

In our particular system there exist five different kind of agents involved in the *Locker Rental Service* that follow the features mentioned in the previous paragraph. Each specific kind of agent has a different specialization area and a task to perform, for which it may need to interact with other agents and places. These agents are the *Mobile Unit (MU) Static* agent, the *Majordomo* agent, the *Locker Rent* agent, the *Locker* agent, and the *Locker Guardian* agent.

The Mobile Unit (MU) Static Agent

At each mobile unit there is a static agent running at any time that the mobile unit is active. This static agent isolates the user's applications from network availability and the specific characteristics of the mobile unit, and is responsible for the administration of the mobile unit's resources. This agent also supports the GUI provided by the ANTARCTICA System to its services users. The GUI is adapted to the mobile unit characteristics. In addition, this static agent creates and launches a single agent from the mobile unit to the fixed network, that is called the *majordomo* agent. This *majordomo* agent remains in the GSN so it is not necessary to create and launch it every time the user requests a service. The static agent at the mobile unit will 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.

The Majordomo Agent

The *majordomo* agent is created at the mobile unit by the static agent and launched to a GSN in 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, this agent remains in the GSN for the time period required, working on behalf of the mobile user even while the mobile device is disconnected.

Each mobile device has its own *majordomo* with the aim of providing adequate services to its owner. Thus, communications of the mobile unit with its outside world are surveyed by a pair of agents, the static agent of the mobile unit 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 mobile unit.

The *majordomo* agent can create other specialist agents that work for it and it distributes and coordinates the tasks among these servants agents, in order to fulfill the requests of its user. Although it would be possible for the *majordomo* agent to perform all the tasks required by its owner, we chose to create specialist agents because it would be difficult for the *majordomo* to use efficiently all the different services offered by the system heeding at the same time the different user requirements.

The *majordomo* agent will 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 mobile unit capabilities, and to optimize the use of the wireless communications, while representing the user in the fixed network.

The Pair of Locker Agents

When the user request to his *majordomo* to use the *Locker Rental Service*, the *majordomo* 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 that we call *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 mobile unit'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 and maintaining services such as a blackboard where the users can leave messages and data for other users. It also handles concurrent data accesses.

On the other hand, 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 in order 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 he can choose which e-mails to read.

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 a database (DB) and monitor the use of the resources.

Finally notice that the implementation of the lockers by a pair of agents collaborating together: the *locker rent* agent and the *locker* agent (the latter agent constitutes the locker itself) allows the management of the lockers 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 by the *guardian* agent 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 over that directory so, it can create, read and write files at that directory. The *locker guardian* agent controls that no *locker* agent access any other directory that its own, and monitors the total disk space the *locker* agent directory is using. No other agent has full access to the storage space. The operative 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. In the case of a shared locker the *guardian* agent provides the *locker* agent with a directory. When a *locker rent* agent arrives to the shared locker, the *guardian* agent creates a new subdirectory inside of the locker directory and notifies the *locker* agent to store there the private data of the user of the newly arrived *locker rent* agent.

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.

3.2 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 [19]. This is the case in our system, since 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 [19]. The central issue is to prevent uncontrolled access. In our system, the only agents that may come from the outside, i.e. are created from classes foreign to the GSNs, are the *majordomos*. On the one hand, it must be ensured that the agents have been programmed, released and signed by a known entity. To authenticate code and its origins, cryptographic techniques can be used, as for example signatures [18]. On the other hand, it must be ensured that the agents represent authorized users. In order to do this, credentials are associated with the agents, according to the identity of their user, access level to the services and data, and access key. The agents credentials are checked against the information the system maintains about the registered users.

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 accounting mechanisms and policies.

4. MOBILITY OF THE LOCKERS

In our proposal, the *majordomo* agent and the private locker that belongs to a mobile user are located at the GSN under whose coverage the mobile unit is situated. This means that when the mobile device 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. Remember that the possibility of moving the locker is one feature that differentiate our proposal from the others and it is naturally supported by the mobile agent technology. The benefit of moving the locker stems for the fact that the cost of the interactions of the two agents (the locker and the majordomo) is reduced, since the agents are located in near-by sites.

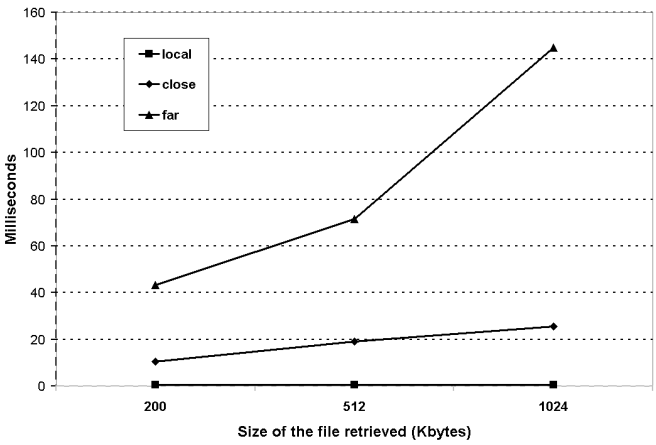

Figure 2: Average time for the interaction between the *majordomo* and its locker when retrieving files stored

Figure 2 shows the average time it takes the *majordomo* agent to retrieve a file of 200 Kbytes, 512 Kbytes or 1024 Kbytes, from a locker situated first locally in the same GSN where the *majordomo* agent is situated, second when they are situated not in the same GSN but in two close ones, and finally, when the *majordomo* and its locker are situated in two far away GSNs [6]. As it was expected, the closer the locker is situated to its *majordomo* agent, the faster this last one can store or retrieve the data stored when serving the user requests or the other agents data and storage needs.

Nevertheless, when deciding to occupy a locker, the user can choose among three main options and specify whether he wants: (a) the locker to always move following his movements, (b) the locker to remain stationary, or (c) the system to decide when to move the locker.

In every case, when the *majordomo* detects his user has moved and is now under the coverage of a different GSN,

[6]For the experiments presented in this paper we used three similar computers playing the role of GSNs. Two of the computers were situated very close, in the same subnetwork, at the University of Zaragoza in the city of Zaragoza, Spain. The third GSN was situated at the University of the Basque Country, in the city of San Sebastian, Spain. The distance between these cities is 270 Km. The two GSNs in Zaragoza were used as the two *close* GSNs, while one of the GSNs at Zaragoza and the GSN at San Sebastian were used as the two *far* GSNs, being situated 10 hops away one from the other. The three GSNs were fixed computers equipped with a 1.7GHz Pentium IV processor, 256 M of RAM and were running Linux Mandrake 8.0 or Linux Debian 2.2. The ANTARCTICA prototype is being implemented using the platform "Aglets Workbench" [8].

it notifies the specialist agents working currently for him that it is going to move, then it moves itself and once at the new GSN it notifies again its specialist agents of its new location.

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 mobile unit is placed at the time the user ask for occupying a locker, and stays there until it is destroyed.

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. When the *locker rent* agent receives the notification of the *majordomo* from its new location, it asks its local *Locker Place guardian* agent for advise and support on the locker moving following the *majordomo*. So, first, the *guardian* agent 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. The negotiation is performed by the local *guardian* agent and the remote *guardian* agent. If the remote *guardian* agent agrees on allocating the locker, then the local *guardian* notifies the requesting *locker rent* agent that the movement is possible. Next, the remote *guardian* creates a new *locker* agent. The two *guardian* agents introduces both *locker* agents (the old and the new one) so they can interchange a copy of the locker contents. Once the locker in the remote place had received all the contents of the locker, (it is actually a duplication of the old locker) the *locker rent* agent leaves the current *Locker Place* and travels to the remote one. On the one hand, when the *guardian* agent of the first *Locker Place* detects that the agent is leaving, it destroys the old locker. On the other hand, when the *guardian* agent of the remote *Locker Place* detects that the *locker rent* agent of an incoming locker is arriving, it introduces the arriving *locker rent* agent to the local *locker* agent that was created for it, and the locker movement is then completed.

Briefly, and not taking into account the time it takes to find out the destination *Locker Place*, the decision of moving or not the locker is taken on the basis of the cost of the movement of the locker (C_{Transf}), considering: the time it takes the duplication of the locker at the new GSN, and the *locker rent* agent trip and its entrance in the new *Locker Place*.

That means, in the computation of C_{Transf}, the following must be considered: (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. In Figure 3 we can observe how much it costs to move a locker depending on its size and location.

However, as Figure 2 shows, having the locker closer to

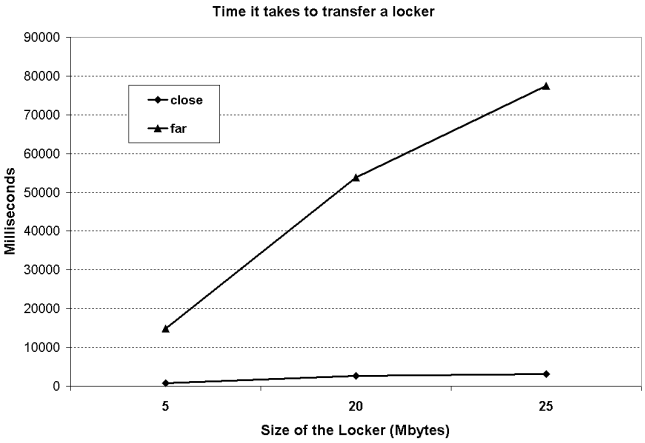

Figure 3: Influence of the distance between GSNs when moving a locker.

the *majordomo* will means a time saving in future interactions between them. The cost of not moving the locker is then the cost of the interaction between the *majordomo* and the locker in the case they are located at different GSNs ($C_{NotMove}$), while the cost of moving the locker (C_{Move}) consists of the cost of the locker movement (C_{Transf}) plus the cost of the interaction between the *majordomo* and the locker when situated in the same GSN or in two close GSNs ($C_{CloseInt}$).

The cost of the interaction[7] 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 the 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.

Without going into details and without considering the cost of interchanging few messages among the agents in order to take the decision of moving or not the locker and coordinate the movement, the costs under consideration are:

[7]By interaction we consider only requests for data, that is, that operation for which the user, or the *majordomo* agent, waits for an answer. The *majordomo* can order update operations on the locker while it continues working in parallel with other tasks.

$$C_{NotMove} = T_{stay} \times N_{Requests} \times C_{Request}\ ,\ where: \\ C_{Request} = C_{FarCm} \times S_{Avg} \quad (1)$$

$$C_{Move} = C_{CloseInt} + C_{Transf}\ ,\ where: \\ C_{CloseInt} = T_{stay} \times N_{Requests} \times C_{NearCm} \times S_{Avg}\ ,\ and \\ C_{Transf} = T_{CrLAg} + S \times T_{Send} + T_{MoveLRAg} \quad (2)$$

According to this, the locker must be transferred if $C_{NotMove}$ is estimated to be greater than C_{Move}, or equivalently, when the ratio C_{Move} / $C_{NotMove}$ is smaller than 1. See formula 3:

$$\frac{C_{Move}}{C_{NotMove}} = \frac{C_{CloseInt}+C_{Transf}}{C_{NotMove}} = \\ \frac{C_{CloseInt}}{C_{NotMove}} + \frac{C_{Transf}}{C_{NotMove}} = \frac{C_{NearCm}}{C_{FarCm}} + \frac{C_{Transf}}{C_{NotMove}} \quad (3)$$

Figure 4 shows the results obtained for the ratio formula 3 when considering a 25 Mbytes locker (containing 128 files of 200 Kbytes each). We can observe that C_{Move} is mainly affected by the time it takes to transfer the locker (about $77s$ in average), since once it is transferred, the interaction between the *majordomo* and the locker is extremely quick. However, $C_{NotMove}$ is mainly affected by C_{FarCm}, which is its slope in the graphs.

The experimental results obtained show that, for having $C_{NotMove}$ and C_{Move} compensating each other, the *majordomo* should interact with its locker at the new location at least for a total amount of data something larger than the total size of the locker, which seems to mean that the locker should rarely move. However, on the one hand, the locker is not moved at the time the user first request something from the locker, but it is moved when the *majordomo* detects the user has moved and after it moves itself to the new GSN. On the other hand, the locker moves on his own while the *majordomo* can continue working and interacting even with the user. That means, whether the locker should be moved or not is decided mainly based on the estimation of the time it is going to take the locker to move (C_{Transf}) and the estimation of how much time is going to pass until the user's next request for data stored in the locker, and if it is going to happen while the locker is moving and so forcing it to wait until the movement is done. Figure 3 shows that the movement of a 25 Mbytes locker took about 77 seconds when moving between the two farthest GSN considered in the experiments. The estimation of C_{Transf} can be computed by the system, while the other time can be estimated by the *majordomo* based on the previous behavior of the user. The *majordomo* could even delay the movement of the locker until the user disconnects for a while, or enters an idle period.

Finally, notice that the possibility of moving a locker is useful not only because the data can be 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.

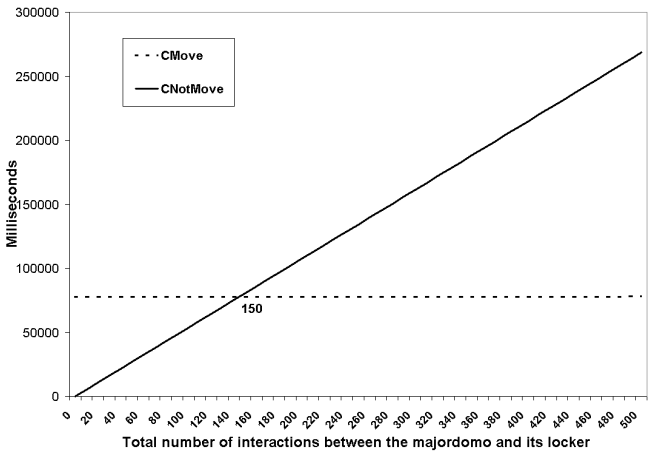

Figure 4: Computation of $C_{NotMove}$ and C_{Move}.

5. IMPROVING PERFORMANCE USING A CACHE

In order to improve the general performance and the *majordomo* response time, we have considered that 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 is feasible because all the data obtained by the agents, or sent to/by the user, actually passes through the *majordomo*. This cache is built and populated with data while the *majordomo* works and continue serving requests from the user. Of course, this cache must be kept small since the *majordomo* can not keep with it large amounts of data. The management of a cache by the *majordomo* introduces an overhead in its operation time since not only has it to access the cache, but also handle cache replacements and maintain cache consistency. However, in general that 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 those data. For the following experiments the *majordomo* was maintaining a cache of a 1 Mbyte maximum size and based on the LRU algorithms. The cache also increases the time it takes the *majordomo* to move since now the agent is more heavy, however the results obtained shows that when the *majordomo* was moving between the two farthest GSNs it took it an average time of less than 1 second when it had no cache, and only an average of less than 4 seconds with a full 1 Mbyte cache.

Using this approach the cache will always move with the *majordomo*, and so, the formulas to determine if the locker must be moved or not are now different.[8] Briefly described and not considering the small cost of checking the cache whenever a data is being requested, the cost of processing a request of data ($C_{Request}(s/du)$), is the cost of retrieving that data from the cache ($C_{AccCache}$) or the cost of retrieving it from the locker ($C_{AccLocker}$), and α is the probability of

[8]Notice that the *majordomo* stores in its cache the data most accessed, this means it does not only stores data that is 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.

finding the data in the *majordomo* cache, i.e. the cache hits.

Therefore, formulas (1) and (2) that represent when the locker must be moved or not are modified. There exists 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. In formula (2) the term C_{Transf} stays the same as before and formulas (1) and (2) are rewritten as follows:

$$\begin{array}{l} C_{NotMove} = T_{stay} \times N_{Requests} \times C_{Request}\ ,\ where: \\ C_{Request} = \alpha \times C_{AccCache} + (1-\alpha) \times C_{AccLocker}\ , \\ C_{AccCache} = T_{AccCache} \times S_{Avg}\ ,\ and \\ C_{AccLocker} = C_{FarCm} \times S_{Avg} \end{array} \quad (4)$$

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

The ratio formula is now:

$$\frac{C_{Move}}{C_{NotMove}} = \frac{C_{CloseInt}+C_{Transf}}{C_{NotMove}} = \frac{C_{CloseInt}}{C_{NotMove}} + \frac{C_{Transf}}{C_{NotMove}} = \frac{\alpha \times T_{AccCache}+(1-\alpha)\times C_{NearCm}}{\alpha \times T_{AccCache}+(1-\alpha)\times C_{FarCm}} + \frac{C_{Transf}}{C_{NotMove}} \quad (6)$$

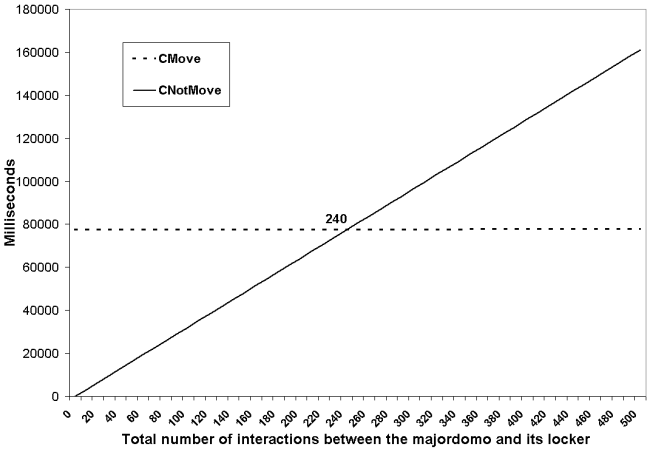

Figure 5: Computation of $C_{NotMove}$ and C_{Move} when $\alpha = 0.4$

Figures 5 and 6 show the experimental results obtained for the ratio formula 6 when considering a 25 Mbytes locker (containing 128 files of 200 Kbytes each), 1 Mbyte of cache and varying parameters: α, and the total number of request or interactions between the *majordomo* and the locker, that is $T_{stay} \times N_{Requests}$ in the formulas. The experiment was performed considering the case of the two farthest GSNs. Figures 5 and 6 show how C_{Move} and $C_{NotMove}$ are affected by α when a cache is used. Notice than the case of not having cache is equivalent to the case in which α equals 0, which correspond to the scenario shown in Figure 4.

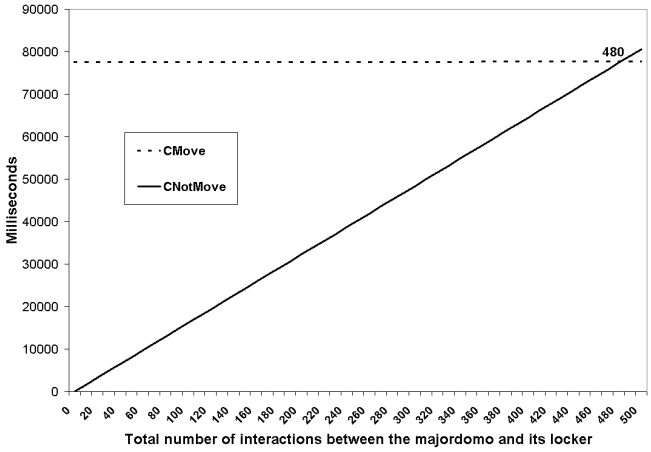

Figure 6: Computation of $C_{NotMove}$ and C_{Move} when $\alpha = 0.7$

Notice that the use of the cache reduces the need of moving the locker, but it also contributes to increase the time the *majordomo* can continue working delaying its interaction with the locker and so giving the locker time to move.

6. SHADOW LOCKERS

Instead of having the *majordomo* agent dealing with a cache another possibility that can be considered is to have only a copy of a small part of the locker following the *majordomo*. This copy could be a kind of hot-locker that will follow the movements of the user and his *majordomo* keeping only the more accessed data but always at the same GSN than the *majordomo*, i.e. the same idea than the cache but having those data stored persistently as in lockers. We call this kind of hot-locker, **shadow locker**, since it does not have the entity or autonomy of a real locker belonging to the user and it can only exists while being related to a real locker. Of course, to retrieve the data stored in the shadow locker will not be as fast as retrieving it from the cache, but while the size of the cache must be kept very small, 1 Mbyte in the experiments, the size of the shadow locker could be bigger, such as for example 5 Mbytes[9], also the *majordomo* will not have the extra work of managing the cache, and the data are kept in persistent storage.

Now, when the user decides to occupy a locker he has a fourth option to choose among, that is to have only a small part of the locker following his movements, the shadow locker. With this option, when the *majordomo* moves to a new location it warns its corresponding *locker rent* agent and it begins a process similar to the locker movement but what is created at the new *Locker Place* is now a shadow locker that receives only a copy of the more accessed data of the locker. From there on, the *majordomo* deals directly with the shadow locker. In terms of data consistency maintenance and data replacement, the shadow locker would work as a

[9]Notice that, in the experiments a 5 Mbytes locker moved in less than one second from one GSN to a close one.

cache, for example applying LRU algorithms, but with the support of the real locker. So, when the *majordomo* requests some data that are not currently in the shadow locker, this one automatically fetches them from the real locker and handles them to the *majordomo*. When the *majordomo* sends some data to be stored, the shadow locker automatically passes them also to the real locker. The shadow locker will be automatically created when the *majordomo* moves away from the locker, transparently to the user, and from there on it will continue following the *majordomo* movements until it happens to came back to where the locker was staying, or until the system can transfers the real locker to the GSN where the *majordomo* is located, and then the shadow locker is destroyed.

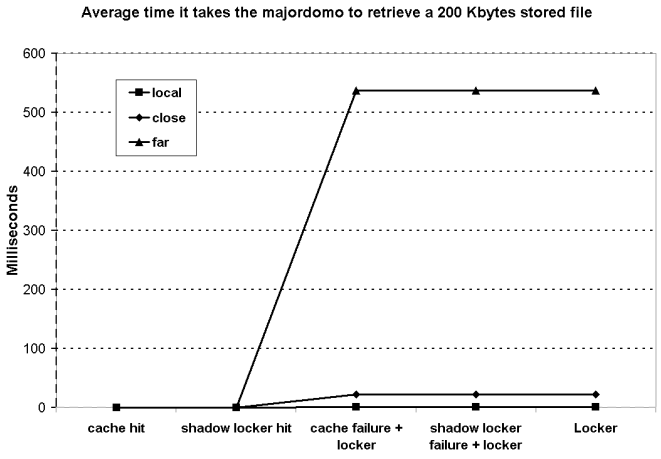

Figure 7: Retrieving a file by all the scenarios considered in this paper

Figure 7 summarizes the average time it took the *majordomo* to retrieve a 200 Kbytes file from the cache (cache hit), or retrieving it from the locker after checking the cache (cache failure), similarly when a shadow locker was being used, and finally when only the locker was used. For the case of using a locker three different locations were considered: local at the same GSN than the *majordomo*, in a different GSN but a close one, and finally in the two farthest GSNs considered in these experiments.

7. CONCLUSIONS

In this paper, we present a new data service for the mobile users: the *Locker Rental Service*. This service allows its users to use persistent storage space located at the fixed network, outside of their mobile device, accessible anywhere-anytime. Besides providing an extension of the storage space of the mobile device, the service gives to its users on the one hand, 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. On the other hand, the service provides lockers mobility keeping the user data always as close to his actual physical location as possible, and so optimizing the cost of accessing the data stored. Experimental results demonstrate the performance improvement obtained moving the whole locker or just small parts containing the most acceded data.

8. REFERENCES

[1] Cyber-Ark Software Ltd. http://www.cyber-ark.com/.

[2] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, October 2001.

[3] Driveway Corporation. http://www.driveway.com/.

[4] Fleet Boston Financial Corporation. http://www.filetrust.com/.

[5] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages.* Springer-Verlag, 1996.

[6] A. Goñi, A. Illarramendi, E. Mena, Y. Villate, and J. Rodriguez. 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, Arizona (USA)*, October 2001.

[7] R. Gray, D. Rus, and D. Kotz. Agent TCL: Targeting de needs of mobile computers. *IEEE Internet Computing*, 1997.

[8] IBM Aglets Workbench - Home Page. http://www.trl.ibm.co.jp/aglets/.

[9] A. Joshi, S. Weerawarana, and E. Houstis. On disconnected browsing of distributed information. In *IEEE RIDE97*, pages 101–107, 1997.

[10] J. Kistler and M. Satyanarayanan. Disconnected operation in the Coda file system. *ACM Transactions on Computer Systems*, 10:3–25, 1992.

[11] 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.

[12] J. Kubiatowicz et al. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2000*, November 2000.

[13] Lexias Inc. http://www.mydigivault.com/.

[14] Milojicic et al. MASIF, The OMG Mobile Agent System Interoperability Facility. In *Proceedings of Mobile Agents '98*, September 1998.

[15] S. Papastavrou, G. Samaras, and E. Pitoura. Mobile agents for WWW distributed database access. In *Proceedings of the International Conference on Data Engineering*, 1999.

[16] E. Pitoura and G. Samaras. *Data Management for Mobile Computing*. Kluwer Academic Publishers, 1998.

[17] 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.

[18] B. Schneier. *Applied Cryptography. Second Edition*. John Wiley & Sons, 1996.

[19] C. Tschudin. Mobile agent security. In *Intelligent Information Agents - Agent based information discovery and management on the Internet*, pages 431–445. Matthias Klusch, Springer-Verlag, 1999.

[20] Y. Villate, A. Illarramendi, and E. Pitoura. Data Lockers: Mobile-Agent Based Middleware for the Security and Availability of Roaming Users Data. In *IFCIS International Conference on Cooperative Information Systems (CoopIS'2000), Springer series of Lecture Notes in Computer Science (LNCS), Eliat (Israel)*, September 2000.

[21] Y. Villate, A. Illarramendi, and E. Pitoura. Keep your data safe and available while roaming. *Accepted for publication on MONET, Special Issue on Pervasive Computing*, 2002.

[22] Y. Villate, E. Pitoura, A. Illarramendi, and A. K. Elmagarmid. Extending the data services of mobile computers by external data lockers. In *Proceedings of the Third International Workshop on Mobility in Databases and Distributed Systems, MDDS'2000*. IEEE Computer Society Press, September 2000.

[23] Xdrive Technologies. http://www.xdrive.com/.

[24] YourZ.com Inc. http://www.myspace.com/.

[25] Zephra Corporation. http://www.securitymadesimple.com/.