Software Agents in the Wireless World - Application to Mobile Services -

Zakaria Maamar$^{\phi}$, Wathiq Mansoor$^{\phi}$, and Qusay H. Mahmoud$^{\varphi}$

$^{\phi}$Software Agents Research Group$_{@ZU}$
College of Information Systems
Zayed University
Dubai, United Arab Emirates
zakaria.maamar,wathiq.mansoor@zu.ac.ae

$^{\varphi}$WITLab
School of Computing Science
Simon Fraser University
Vancouver, Canada
qmahmoud@cs.sfu.ca

Abstract The paper discusses mobile services, denoted by m-services, in the wireless world. M-services are viewed as an extension to the e-services paradigm. The wireless world has its own features that make it different from the wired world. For instance, new communication means need to be used, new user-friendly services need to be suggested, and new types of computing resources need to be involved. The paper also discusses the use of software agents in addressing the following issues: how are m-services discovered in an open wireless environment? How are m-services combined together? And how are m-services made executable on wireless devices?

1 Introduction

According to [3], the Internet is going through several major changes. Indeed, the Internet is becoming a vehicle of services rather than just a repository of information. Currently, several organizations are making their services accessible over the web. Usually, e-services denote this kind of services. Besides the new role of the Internet, we expect that more and more e-services will be offered to people who use wireless devices to conduct their operations. Further, we expect that more and more wireless devices will be enhanced with new advanced computing resources, which will enable them in the near future to become mobile offices. We are all witnessing the tremendous growth in the development and use of wireless devices across the world. Unfortunately, this growth is accompanied with its set of problems. For example, wireless devices are strictly bound to their batteries for operation. In order to support wireless devices-oriented users[1], we aim at working on a new generation of e-services that will be specially designed and developed to fit wireless devices. M-services denote this new generation of e-services.

There exist several types of wireless devices, varying from mobile phones to Personal Digital Assistants (PDAs). These devices present several shortcomings that make their use requires specific arrangements. For instance, the services to

[1] Users who use wireless devices to carry out operations.

be offered need to be tailored to each device individually. Unfortunately, this goes against the e-services platform independence principle. Thus, leveraging e-services into m-services is a necessity. M-services will have to consider the characteristics of the wireless devices on which they will be running. In fact, we advocate that m-services will be fetched on-demand from provider sites to users' wireless devices.

Having pre-installed services on users' wireless devices is an option that could be considered. However, this option would have worked for a small number of users where looking for the lowest common denominator between them is feasible. In an open environment with different users and needs, looking for the common denominator is not trivial. Therefore, on-demand delivery of services to wireless devices is more appropriate than pre-installing services. Enabling the downloading of dynamically composed services is among the approaches to follow for systems designed to the wireless world.

Section *2* motivates our work on m-services. Section *3* defines mobile computing and e-services. Section *4* presents our system of m-services. Section *5* describes our system implementation. Section *6* presents related work. Section *7* concludes the paper. It is made clear at that level that issues such as negotiation and coordination, while important, do not fall in the scope of this paper.

2 Motivating scenario

To motivate our work, we consider the following fictive scenario. A sales representative working for a software company; it is based in country X but serves customers from over the world. Since the representative is most of the time on the move visiting customers, he is equipped with a PDA, which provides basic office tools. Moreover, the representative uses the PDA to record the deals he made with customers. It may happen that in preparation for an important meeting, the representative has to carry out specific operations such as simulating the consequences of this meeting on the company stock shares. Unfortunately, before he leaves his office this time he did not download the appropriate software applications into his PDA. Consequently, he decides to proceed as follows. First, he sends a "wireless" request to the company server. According to the workload of the server, one of the following situations occurs.

Server not "busy" - it accepts the request and processes it. To this end, the representative has to send "wirelessly" the required data. Later on, the server transmits "wirelessly" results to the representative's PDA. If the representative has known before that the server was not busy, he could have attached the required data to the request he submitted in the first time. Unfortunately, this is difficult to predict.

Server "busy" - it suggests two alternatives to the representative. The first one is to put his request in a waiting list. The second one is to send the required applications to his PDA for execution. These applications transfer has to follow specific rules, as we will discuss it.

In our work, we are interested in situation *Server "busy"* - second alternative, where a wireless device will be acting as a computing platform to the applications to be received from the server. We consider these applications as m-services.

3 Background

Mobile Computing refers to systems in which computational components, either hardware or software, change locations in a physical environment. Moving from a location to another is due to several reasons: component miniaturization, wireless networks, and mobile-code programming languages. Kinds of mobile work are as follows [15]: hardware mobility, software mobility, and combined mobility. A code that is downloaded from a web site to a user's mobile phone combines both hardware and software mobility.

E-service is an application component provided by an organization in order to be assembled and re-used in a distributed, Internet-based environment [12]. A component is considered as an e-service if it is: 1) independent as much as possible from specific platforms and computing paradigms; 2) developed mainly for inter-organizational situations rather than for intra-organizational situations only; and 3) easily composable with other e-services.

4 System of m-services

4.1 M-services

The rationale of designing and developing m-services[2] is to offer new opportunities to wireless devices-oriented users. It happens that these users postpone their operations because they lack appropriate facilities on their devices. Therefore, it becomes important to support such users by allowing them: 1) to search for additional facilities, when needed, 2) to fetch these facilities to their wireless devices, and 3) to conduct the operations 1) and 2) in a transparent way. A solution to 1) consists of creating brokering mechanisms. A solution to 2) consists of using wireless communication mechanisms. Finally, a solution to 3) consists of using Software Agents (SAs) [6].

We consider an application component as an m-service if it is: transportable through wireless networks; flexible in term of composition with other m-services; adaptable according to the wireless devices' computing characteristics; and executable on wireless devices.

4.2 Architecture

In our work, brokering mechanisms and SAs are considered in the design and development of a system offering m-services to wireless devices-oriented users. This system's features are:

[2] We'll be using m-service and service in an interchange way.

– Develop three types of SAs: user-agent, provider-agent, and device-agent. The first type is associated with users of m-services. The second and third types are associated with providers of m-services.
– Create a software platform, called Meeting Infrastructure (MI), that will be headed by a supervisor-agent. This MI has a brokering role [10].
– Develop two types of delegates: provider-delegate and user-delegate. Delegates interact respectively on behalf of user-agents and provider-agents in the MI.
– Develop storage servers that save the sequence of m-services to be sent to wireless devices for execution. Storage servers are spread across networks and storage-agents are responsible of them.

Figure 1 illustrates the architecture of our system. It is decomposed into four parts: user, provider, MI, and storage. The MI and storage parts are linked to the user part in a wireless way. Meanwhile, the MI and storage parts are linked to the provider-part in a wired way.

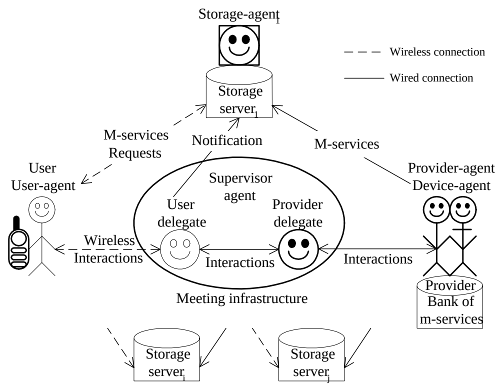


Figure1. Architecture of m-services system

The *user part* consists of users and user-agents. User-agents act on behalf of users; they accept their needs, convert them into requests, and submit them to user-delegates. The MI supervisor creates user-delegates. To satisfy users' requests, user-delegates interact with provider-delegates.

The provider-part consists of providers, provider-agents, and device-agents. Provider-agents act on behalf of providers of m-services; they advertise their m-services to user-delegates, through provider-delegates. In addition, they monitor the behavior of providers in case of new m-services are offered, and thus need to be announced. In Figure 1, m-services are gathered into a bank on which provider-agents and device-agents are located. Provider-agents create provider-delegates. In the provider-part, device-agents support provider-agents' work. The role of device-agents is to wrap m-services before they are sent to users' wireless devices for execution. The rationale of having device-agents is to consider the differences that exist between wireless devices, e.g. screen dimension and processor speed.

The *MI part* is a software platform on which user-delegates and provider-delegates interact in a local and secure environment [9]. In an open environment, most of the interactions between requesters of services and providers of services are conducted through third parties, known as brokers. Despite its important role, a broker could easily become a bottleneck. To overcome this problem, requesters and providers may have to bypass the broker. They need a common environment in which they meet and interact directly. The MI plays the role of this environment. In our system, the MI supervisor-agent has several responsibilities, among them: it monitors the interactions that occur within the MI; it makes the MI a safe environment; and it directs user-delegates towards the provider-delegates that interest them.

The storage part receives the sequence of m-services that will be sent to users for execution. According to our system execution's principles, m-services are sent to a user's wireless device one by one. Several advantages are obtained from the use of storage-servers. A user-agent does not have to deal with several providers. Its point of contact for getting the m-services is the storage-agent. The same thing applies to device-agents that will be interacting with few storage-agents instead of several user-agents. Security is increased for both users and providers. Storage-servers are independent platforms where security control are conducted.

4.3 Software agents

User-oriented components *User-agent* resides in a wireless device. First, the user communicates with his user-agent to arrange requests. After submitting them to the user-delegate, the user-agent goes into a standby mode and waits for notifications from its delegate. Notifications concern the sequence of m-services that satisfies the user's requests. Before executing them on user's device, m-services are put in a storage server. The MI supervisor-agent suggests to user-delegate the storage server to be used considering for example the server's location. To download the m-services one at a time from the storage server to the user's wireless device, the user-agent communicates with the storage-agent. The user-agent keeps track of the execution of an m-service, before it asks the storage-agent to submit the next m-service. When it is received, the executed service is deleted from the wireless device platform. Finally, the user-agent informs the user about the completed requests.

If this is the first time that a user-agent submits requests, the supervisor-agent creates a user-delegate in the MI to be associated with that user-agent (cf. Table 1). For the next requests, the user-agent interacts directly with the user-delegate (cf. Table 2). Table 1 and Table 2 use "submit", "reply", and "notify" performatives illustrating the interactions that take place between user-agents and supervisor-agent and between user-agents and delegate-agents. In Table 1, "submit" performative has five fields among them "Device"; it specifies the characteristics of the wireless device the user is using. The device-agent considers these characteristics in its process of wrapping m-services. In Table 2, "submit" performative occurs once a user-agent knows the delegate-agent to which it has been associated.

Table1. Interactions user-agent/supervisor-agent

Submit(Id-submit: id$_1$, From: user-agent$_1$, To: supervisor-agent$_1$, Content: request$_1$, Device: (Brand: brand$_1$, Type: type$_1$, Screen dim.: dim$_1$))
Reply(Id-reply: id$_1$, From: supervisor-agent$_1$, To: user-agent$_1$ (In-reply-to: id$_1$), Delegate: (Id: user-delegate$_1$, Contact: user-delegate$_1$@...), Storage-server: (Id: storage-agent$_1$, Contact: storage-agent$_1$@...))

Table2. Interactions user-agent/user-delegate

Submit(Id-submit: id$_2$, From: user-agent$_1$, To: user-delegate$_1$, Content: request$_2$, Device: ())
Notify(Id-notify: id$_1$, From: user-delegate$_1$, To: user-agent$_1$ (In-reply-to: id$_2$), Content: sequence$_1$/m-service$_{1,2}$)

User-delegate resides in the MI, acting on behalf of user-agent. The user-delegate receives the user's requests from the user-agent (cf. Table 2). Afterwards, it interacts with provider-delegates. The purpose is to match user's requests with providers' m-services. In case there is a match (we assume that there is always a match), the user-delegate designs the sequence of m-services to be included in satisfying the user's requests. Information about this sequence is sent to the storage-agent (cf. Table 3). The purpose is to make the storage-agent ready to receive m-services from device-agents. Furthermore, the user-delegate notifies the user-agent about the sequence of m-services it has established (cf. Table 2). In Table 3, two relevant fields of "to-get-ready" performative are important. "Destination" field corresponds to the user-agent for which the sequence of m-services has been designed. "M-services" field corresponds to the m-services the device-agent has to submit. To set up a sequence, the storage-agent knows the m-service that comes before and after the m-services to be submitted by a device-agent. Instead of creating a user-delegate on a wireless-device platform and shipping it to the MI, we suggested to undertake this operation in the MI for two main reasons: even if we expect a big improvement in wireless devices' resources, those resources have to be used in a "rationale" way; and the wireless connection that transfers the user-delegate is avoided.

Provider-oriented components *Provider-agent* resides in provider's site, running on top of its resources. A part of these resources correspond to m-services.

Table3. Interactions user-delegate/storage-agent

To-get-ready(Id-to-get-ready: id_4, From: user-delegate_1, To: storage-agent_1, Sequence: sequence_1, Destination: user-agent_1@..., M-services: (From: device-agent_i, Before: 0[m-service_h]1, Id: 1[m-service_i]n, After: 0[m-service_j]1))

Provider-delegates broadcast m-services to user-agents, through user-delegates. The provider-agent is in constant interactions with its provider-delegate (cf. Table 4). For instance: it communicates to the provider-delegate the negotiation strategy it has to follow with user-delegates.

Table4. Interactions provider-agent/provider-delegate

Submit(Id-submit: id_6, From: provider-agent_1, To: $\text{provider-delegate}_1$, Content: strategy_6/m-service_6, Device: ())
Inform(Id-inform: id_1, From: $\text{provider-delegate}_1$, To: provider-agent_1, Agreement: (Sequence: sequence_1, With: user-agent_1, Service: 1[Id: m-service_i]n), Storage-server: (Id: storage-agent_1, Contact: storage-agent_1@...), Device: (Brand: brand_1, Type: type_1, Screen dim.: dim_1))

Device-agent resides in provider's site. Its responsibility is to wrap m-services according to the devices to which they will be sent for execution. Initially, these m-services are sent to storage servers. The provider-agent has already submitted the contact details of the storage-server to the device agent. We recall that the user-delegate has informed the storage-agent of that storage server about the m-services it will receive (cf. Table 3). Double-checking the information that user-delegates and provider-delegates forward to a storage-agent guarantees more security to our system.

Provider-delegate resides in the MI; it acts on behalf of provider-agent. In our system, the provider-delegate is responsible for interacting with user-delegates, regarding the m-services it offers. In addition, it interacts with its provider-agent for notification purposes. Notifications are then forwarded to device-agent for consideration. We recall that the provider-agent creates a provider-delegate and transfers it to the MI.

MI-oriented components *Supervisor-agent* resides in the MI and has several responsibilities: it supervises the operations that occur in the MI; it mediates in case of conflicts between user-delegates and provider-delegates; it sets-up user-delegates and assigns them to user-agents (cf. Table 1); it checks provider-delegates identity when they arrive from provider sites; and it suggests to user-delegates the storage server to be used.

User-delegate and *provider-delegate* are explained above.

Storage-oriented components *Storage-agent* resides on top of storage server. The purpose of such a server is to save the m-services to be sent to wireless devices for execution. According to the information on the sequence of m-services it has received from the user-delegate (cf. Table 3, "before" and "after" fields), the user-delegate arranges the sequence as the m-services start arriving from providers. As soon as this sequence is completed, it informs the user-agent in order to get prepared (cf. Table 5).

Table5. Interactions device-agent/user-agent

Get-prepared(Id-get-prepared: id_7, From: storage-$agent_1$, To: user-$agent_1$, Sequence: $sequence_1$, Status: ready)

Based on the requests it receives from user-agent, the storage-agent pushes the m-services one at a time (cf. Table 6). These m-services are ready for execution. The deletion of m-services from the storage-servers as well as from wireless devices follows specific reliability rules (cf. Figure 3).

Table6. Interactions user-agent/storage-agent

Request(Id-request: id_{24}, From: user-$agent_1$, To: storage-$agent_1$, Sequence: $sequence_1$, M-service: ?m-$service_1$)
Push(Id-push: id_1, From: storage-$agent_1$, To: user-$agent_1$ (In-reply-to: id_{24}), Already submitted: 3, Remained: 2, Attachment: m-$service_1$)

4.4 Operating mode

The operating mode of our system consists of five stages: agentification, identification, correspondence, notification, and realization (cf. Figure 2).

Agentification stage purpose is to set up the different infrastructures and agents that constitute our system. User-agents are established at the user level. Provider-agents and device-agents are established at the provider level. Last but not least, the meeting infrastructure and storage servers, including their storage-agents, are created.

Identification stage purpose is to inform the MI supervisor-agent about the existence of users and providers who are interested in our system. At the agentification stage, user-agents and provider-agents are respectively installed on top of users' wireless devices and providers' resources. The outcome of the identification stage is the creation of user-delegates and the reception of provider-delegates coming from provider-sites. Creation and reception occur in the MI. Provider-agents inform the supervisor-agent about their readiness to submit their provider-delegates to the MI. User-agents inform the supervisor-agent about the users' requests they would like to submit.

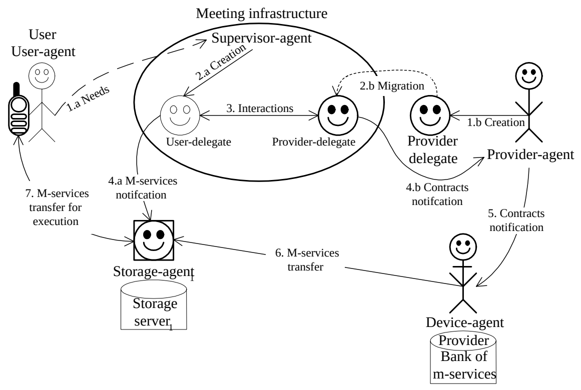


Figure2. Operating of m-services system

Correspondence stage purpose is to enable user-delegates and provider-delegates to get together. User-delegates have requests to satisfy and provider-delegates have services to offer. First, a user-delegate searches for the provider-delegates that have the services it needs. Two approaches exist (cf. Table 7 and Table 8): a/ The user-delegate asks the supervisor-agent to suggest a list of provider-delegates that have the services it needs. Here, the supervisor-agent plays the role of a recommender. b/ The user-delegate requests from the supervisor-agent the contact details of all the provider-delegates that exist in the MI. Table 8 lists the advantages and disadvantages of both approaches.

Table7. Interactions user-delegate/supervisor-agent

Ask-for(Id-ask-for: id_{32}, From: user-delegate1_1, To: supervisor-agent_1, Content: 1[service_i,?provider_i]n)
Recommend(Id-recommend: id_1, From: supervisor-agent_1, To: user-delegate1_1 (In-reply-to: id_{32}), Content: a/ Provider-$\text{delegate}_{1,4}$ $\oplus$ b/ Provider-delegates)

Independently of the approach of searching for delegate-providers, the user-delegate submits its needs of services to selected provider-delegates after interactions. Based on different parameters, such as workload and commitments, provider-delegates answer the user-delegate. At this stage of our work, we assume that providers do not have services in common. Consequently, there is no need for a user-delegate to look for the best service (negotiation for a service is 1 user with 1 provider)[3]. Once the user-delegate and provider-delegates agree

[3] This is not a shortcoming of our system. It is part of our current work to consider 1:N negotiation.

Table8. Advantages and disadvantages of searching for provider-delegates approaches

Advantages	Disadvantages
a/ Provider-delegates are targeted in advance. Less interaction messages between user-delegates and provider-delegates.	a/ Supervisor-agent becomes a bottleneck. Supervisor-agent's recommendations could not satisfy completely user-delegates. User-delegates have a narrow perception of the MI content.
b/ User-delegates have a wide perception of the MI content. More freedom is given to user-delegates.	b/ More messages to discover what provider-delegates offer.

upon the m-services to use, notifications are sent to different recipients. This will be explained in the notification stage.

Notification stage purpose is to inform different agents about user-delegates' and provider-delegates' agreements.

Regarding the user-delegate, it is in charge of the following operations: informs the user-agent about the sequence of m-services it has established to satisfy its user's request (cf. Table 2); and informs the storage-agent about the sequence of services it will receive from different device-agents (cf. Table 3).

Regarding the provider-delegate, it notifies the provider-agent about its agreements with a user-delegate (cf. Table 4). We recall that provider-agents do not reject any agreements.

Based on the information it has received from its provider-delegate, the provider-agent forwards them to the device-agent. This information concerns the m-services that are involved and the storage-server that is used. Among the actions the device-agent undertakes we cite submitting the m-services to the storage-agent of the storage-server.

Realization stage purpose is to execute the sequence of m-services that the user-delegate has designed. User-agent and storage-agent are the participants of this stage. We recall that the user-delegate has already informed the storage-agent about the m-services it will receive from device-agents (cf. Table 3). Before the user-agent starts asking the storage-agent for the m-services it has, it waits for a notification message from the storage-agent mentioning that the sequence is ready to be submitted for execution.

In the realization phase, reliability aspect has been taken into account. Our approach considers a storage-server as a back up server for the m-services. When a storage-agent sends an m-service to a user-agent, the storage-agent keeps a copy of this service at its level. It deletes it when the user-agent asks for the m-service that follows the one it has received. For the last m-service of a sequence, the user-agent sends an acknowledgement message to the storage-agent so it deletes that m-service (cf. Figure 3).

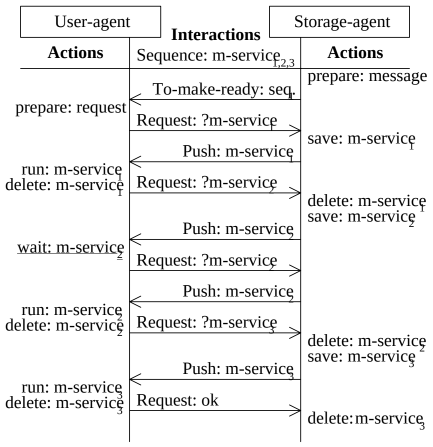


Figure3. Interaction diagram of realization phase

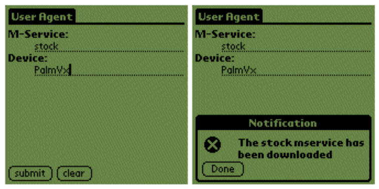
5 Implementation

An implementation of the MI has already been undertaken in [9]. The prototype uses SUN's JavaSpaces technology [5] to implement a repository that holds providers' m-services advertisement. JavaSpaces is an implementation of Linda [8] that provides a simple, fast, and a unified mechanism for sharing, coordinating, and communicating distributed resources, services, and objects across the network.

The MI is implemented as a set of three modules. The first module is the management and installation, which is responsible for receiving delegates from their original systems and installing them. All the security operations are undertaken in this module. The second module is the interaction module that supports the communications that take place between user-delegates and provider-delegates. Finally, the third module is the advertisement and consulting module, which is supported by JavaSpaces, deals with advertising services and identifying the services that are required for satisfying specific needs. In addition, the advertisement and consulting module supports the subscription process to user-delegates. For instance, if a user-delegates is interested in a specific event, then it can be notified when this event occurs. JavaSpaces provides this notification operation.

Provider-agents and their delegates and user-delegates are implemented using Voyager [14]. Regarding the user-agent, it is mainly an interface-agent that feeds the user-delegate with requests. This agent is implemented using J2ME [4], which is a Java edition that runs on wireless handheld devices with constrained resources. A snapshot of the user-agent prototype is shown in Figure 4.

At this stage of our implementation, we simulate the situation where a user can enter the m-services he requires. In this example, the user is requesting a stock m-service that can be used to simulate stock operations. Once the stock



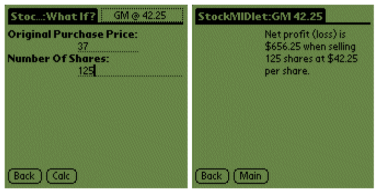
M-Service:
stock
Device:
PalmVx

M-Service:
stock
Device:
PalmVx

The stock mservice has been downloaded

Figure4. User-agent screen mockup

m-service has been downloaded the user will be notified, as shown in Figure 4, so that she can start using it as shown in Figure 5.



Original Purchase Price:
37
Number Of Shares:
125

Net profit (loss) is $656.25 when selling 125 shares at $42.25 per share.

Figure5. M-service execution screen mockup

6 Related work

There exist several projects that study how wireless devices could change our way of doing business [1,10,11,13]. In HP Laboratories, [10] worked on delivering Internet services to wireless users. This work was conducted under the project Ψ for Pervasive Services Infrastructure (PSI). The Ψ vision is "any service to any client (anytime, anywhere)". The project investigated how offloading parts of applications to mid-point servers can enable and enhance service execution on a resource-constrained device. In our work, we are interested in the same issues. Furthermore, we are interested in supporting users in their search for m-services in an open environment. In [11] the Odyssey project provides system support for mobile and adaptive applications. It defines a platform for adaptive mobile data access on which different applications, such as a web browser, a video player, and a speech recognition, could run on top. The Odyssey's approach is to adjust the quality of accessed data to match available resources. In our system, we considered adaptability at two different levels: m-services are wrapped according to the wireless devices' characteristics; and user-agents request m-services from storage agents according to the resources that are available on their users' devices.

Another related project to our work is Ninja [13]. It aimed at suggesting new types of robust and scalable distributed Internet services. Ninja's objective is to meet the requirements of an emerging class of extremely heterogeneous devices that would access these services in a transparent way. In Ninja, the proposed architecture considered four elements: bases, units, active proxies, and paths. Comparable to our sequence of m-services, a path is an abstraction through which units, services, and active proxies are composed. Proxies are transformational intermediaries put between devices and services to shield them from each other. Ninja proxies are similar to our device-agents. In addition, Ninja suggested a Service Discovery Service (SDS) for two reasons: enable services to announce their presence and enable users and programs to locate these announced services. Similar to the SDS, the MI has a brokering role. Moreover, the MI facilitates the direct interactions between providers and users. In fact, the MI avoids bottleneck situations and ensures a better security.

7 Conclusion

In this paper, we presented our work on m-services. M-services are considered as an extension to e-services. Users are more and more relying on their wireless devices to complete their daily operations. Therefore, new facilities should be provided. Our work aims at achieving "anywhere and anytime" paradigm. This paradigm could be summarized by [2] who stated "*How to make a set of heterogeneous services implemented and provided by different providers available for roaming users as an integrated package, with the same "look and feel" across different networks, and from different terminals*".

Several aspects are still under investigation, among them security [7] and scalability. Our suggestion for security consists of enhancing the storage-agent with security mechanisms that could be generic; in the sense that these mechanisms will be applied to all m-services despite their origin (i.e. device-agent) and destination (i.e. user-agent). Specific security procedures that answer each user's priorities should be done at the wireless device level. Since the resources on wireless devices should be used in a rationale way, our future work is based on trust between user-agents and storage-agents. For instance, if a user-agent had the opportunity to deal with the same storage-agent several times and based on its previous experiences, the user-agent could entrust to this storage-agent its specific security procedures.

Regarding scalability, the architecture of the m-services system we suggested in Figure 1 is highly scalable. The system can be extended, at a reasonable cost, as the demand for the m-services it provides increases. This can be achieved by adding more meeting infrastructures to a wireless carrier network. The bank of m-services and the storage-servers can be replicated across the network. This allows us to avoid the performance bottleneck that would arise if a single MI has to handle all client requests.

References

1. S. Case, N. Azarmi, M. Thint, and T. Ohtani. Enhancing e-communities with agent-based systems. *IEEE Computer*, July 2001.
2. L. Esmahi, R. Impey, and R. Liscano. An architecture for providing mobile integrated services for roaming users. In *Proceedings Proceedings of MICON*, Ottawa, 2000.
3. M.C. Fauvet, M. Dumas, B. Benatallah, and H. Paik. Peer-to-peer traced execution of composite services. In *Proceedings of the International Workshop on Technologies for E-Services (TES'2001) held in conjunction with VLDB'2001*, Italy, 2001.
4. J2ME. http://java.sun.com/j2me, Visited Nov. 2001.
5. JavaSpaces. http://java.sun.com/products/javaspacesl, Visited Nov. 2001.
6. N. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1), 1998.
7. S. Key Miller. Facing the challenge of wireless security. *IEEE Computer*, July 2001.
8. Linda. http://www.cs.yale.edu/Linda/linda.html, Visited Nov. 2001.
9. Z. Maamar, E. Dorion, and C. Daigle. Towards virtual marketplaces for e-commerce. *Communications of the ACM*, 44(12), December 2001.
10. D. Milojicic, A. Messer, p. Bernadat, I. Greenberg, G. Fu, O. Spinczyk, D. Beuche, and w. Schroder-Preikschart. ψ - pervasive services infrastructure. Technical report, HP Technical Report HPL-2001-87, HP Laboratories Palo Alto, 2001.
11. B.D. Noble, M. Satyanarayanan, D. Narayanan, J.E. Tilton, J. Flinn, and K.R. Walker. Agile application-aware adaptation or mobility. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, France, 1997.
12. B. Pernici and M. Mecella. Designing components for e-services. In *Proceedings of the VLDB Workshop on Technologies for E-Services (VLDB-TES 2000)*, Egypt, 2000.
13. The Ninja project. http://ninja.cs.berkeley.edu, Visited September 2001.
14. Voyager. http://www.objectspace.com/products/voyager, Visited Nov. 2001.
15. A. I. Wand and L. Chunnian. Process support for mobile work across heterogeneous systems. Technical report, Department of Information Science, Norwegian University of Science and Technology, Norway, 2001.