A Mobile Agent Infrastructure for the Mobility Support

Paolo Bellavista
Dip. Elettronica, Informatica e Sistemistica – Università di Bologna
Viale Risorgimento, 2
Bologna - Italy
Ph.: +39-051-2093087
pbellavista@deis.unibo.it

Antonio Corradi
Dip. Elettronica, Informatica e Sistemistica – Università di Bologna
Viale Risorgimento, 2
Bologna - Italy
Ph.: +39-051-2093083
acorradi@deis.unibo.it

Cesare Stefanelli
Dipartimento di Ingegneria
Università di Ferrara
Via Saragat, 1
Ferrara - Italy
Ph.: +39-0532-768602
cstefanelli@ing.unife.it

ABSTRACT

The mobility of terminals and users is a crucial issue in the open global system represented by the Internet. Supporting terminal and user mobility requires a middleware infrastructure capable of efficiently answering the needs of scalable resource discovery, of security and interoperability, of Quality of Service (QoS) monitoring and adaptation. Some proposals address mobility at the network level and some others focus on application-level solutions. By following the latter approach, the paper proposes the adoption of the mobile agent technology to model and implement mobility. In particular, the paper concentrates on the components and modules implemented in the SOMA mobile agent programming framework to specifically support terminal and user mobility. The SOMA tracing and discovery system extends the SOMA basic naming service to identify and keep track of all mobile entities in the environment. The SOMA QoS adaptation support exploits the functionality of the SOMA monitoring tools and permits to dynamically adjust service provision in response to the changing network and nodes conditions. These features are integrated in a mobility add-on module that pursues also the goals of security and interoperability when moving users and roaming terminals.

Keywords

Mobile and Nomadic Computing, Mobile Agents, Mobile Tracing, Resource Discovery, QoS Adaptation.

1 INTRODUCTION

Several proposals in the area of mobility are directed towards the extension of the traditional Internet architecture to accommodate portable computing devices and to support user roaming. In particular, terminal mobility permits mobile devices to roam in the global system while maintaining the same ability of service access and provision, and user mobility provides users a uniform vision of their preferred working environment independently of their current points of attachment [20, 26].

The mobility support requires a flexible distributed infrastructure that is capable not only of identifying and tracing any mobile entity (i.e. terminals, users and software components), but also of adapting the system to dynamically evolving network conditions. In addition, mobility in open environments focuses on the necessity of both granting an adequate security level and overcoming systems heterogeneity. On the one hand, the openness of the system and the potential presence of malicious intruders require preventing attacks and providing suitable security tools. On the other hand, the variety of devices, operations and systems forces to consider interoperability issues by introducing practices and methodologies to define standards and to integrate legacy components.

We claim that providing support to mobility can take advantage of a middleware solution that covers several different layers of abstraction. Although network and transport protocols for mobile hosts can transparently maintain network connectivity, they lack expressive capacity when working at the application level. At the application level, typical services require resource management, naming, security and interoperability, and these services call for distributed and coordinated infrastructures. In this area, many proposals have recognized the central role of mobile code, because code mobility significantly simplifies the dynamic upgrade of hosting environments, depending on the evolution of the system and with no need to suspend service provision [22, 23, 24, 28]. This extendibility has mainly motivated the increasing interest in new mobile code technologies, and, in particular, in the Mobile Agent (MA) one, which allows the migration of computing entities together with their state and while in execution.

This paper presents the distributed infrastructure of the **SOMA**[*] (**S**ecure and **O**pen **M**obile **A**gents) programming environment, and particularly focuses on its mobility add-on module. SOMA allows all entities to move autonomously and to adapt to the current system situation, to the current position of the involved users, and to the current characteristics of their points of attachment to the network. The SOMA mobility support is capable of recognizing the dis/connection of a mobile terminal and of adapting to its current position by reconfiguring the needed services and resources. From the point of view of user mobility, the SOMA mobility infrastructure permits to recognize dynamic dis/connections of any identified user and to rearrange the service infrastructure accordingly.

The mobility add-on module consists of two basic components: the tracing and discovery service, and the Quality of Service (QoS) adaptation service. The SOMA tracing and discovery service identifies and keeps track of any mobile entity in the environment; in addition, it provides a resource discovery functionality that allows mobile entities to interrogate their current hosting environment in order to find and exploit locally available resources. The SOMA QoS adaptation service is basic for the mobility support since the nomadicity of users and devices forces to consider rapid changes of resource availability in an absolutely unpredictable manner. The QoS adaptation permits to dynamically modify the service quality offered, depending on the current properties of the involved terminals, of the user desiderata, and of the network and intermediate nodes that take part in service provision.

The paper is structured as follows. Section 2 motivates the adoption of mobile agents as an enabling technology to provide the support for mobility. Section 3 describes the general organization of the SOMA programming environment, while Section 4 gives details about the SOMA mobility add-on. Related work sets our project in the context of current research on terminal and user mobility. Concluding remarks follow.

2 MOBILE AGENT TECHNOLOGIES FOR MOBILITY

Several research projects propose mobile agents as a promising mobile code technology to support enhanced user and terminal mobility [4, 22, 23, 24, 28]. In fact, the requirements induced by mobility are very similar to the ones addressed by the MA technology, and it is not only convenient but also natural to extend the MA infrastructure to support mobility.

Any infrastructure for the mobility support should take into account the specific network scenario where mobility takes place. Mobile users have often to work in critical network conditions, especially in the case of terminal mobility. Wireless connections, for example, impose constraints on the available bandwidth and on the reliability of communications, thus requiring to minimize the expensive connection time; mobile users should be capable of starting computations by launching them in the network without having to rely on stable connections for the duration of the requested operation. Several intrinsic properties of the MA technology, such as mobility and autonomy, represent solutions to the intrinsic problems of this scenario. Agent mobility permits the optimization of the connection time required between mobile terminals and the network [24]. In addition, autonomy permits asynchronicity between user actions and MA-performed tasks. Agents can operate also when the user is temporarily disconnected, and, only at the completion of their duty, they wait for the user connection to yield back results. Differently from a traditional client/server model, stable connection is needed only for the time required to inject the agent into the network and to receive the service results [23].

Mobility also stresses interoperability issues, because terminals tend to move to non completely known environments where they interact with locally available resources. This requires a local directory service to discover available resources [32] and standard interfaces to overcome possible heterogeneity of implementations. Agents have already shown their capacity of acting as proxy to encapsulate resources and legacy systems, thus providing standardized interfaces to service components [5]. Moreover, mobility introduces additional security problems and forces to reconsider solutions for authentication of mobile users and terminals, for authorization in accessing system resources, and for granting secrecy and integrity when needed in communications. Security tools are usually available at the application level and some MA infrastructures already provide mechanisms and policies to grant the needed level of security. For instance, MA systems easily integrate with public key infrastructures that permit distributed authentication of mobile users and terminals.

Finally, a further relevant property of the MA paradigm for the realization of mobility infrastructures is location-awareness. Mobile agents tend to maintain full visibility of the underlying system resources and can propagate this visibility at the service level. Location-awareness is a basic property for application-level optimization of the usage of available resources [8]. Only a mobility infrastructure can offer to applications the expressive capacity of adapting to the current network environment, by trading off between volume of transferred data (and its granted quality of service) and requested time. For instance, a support for terminal mobility must react to any change in the device that a mobile user is currently using (e.g. from a laptop to a light PDA), to better adapt its demands to the network situation (e.g. to include/discard attachments in the download of e-mail messages).

3 THE SOMA INFRASTRUCTURE

We have concentrated our efforts on the design and the implementation of an MA-based integrated infrastructure to support terminal and user mobility. The infrastructure is based on the SOMA programming framework and is implemented in the JAVA language [15].

The SOMA programming environment offers to mobile entities (i.e. agents, terminals and users) a hierarchy of locality abstractions to model and describe any kind of open and global distributed system, ranging from simple LANs to the Internet, as depicted in Figure 1. Any node owns at least one *place* that constitutes the agent execution environment. A specialization of the place abstraction, called *mobile place*, models nomadic terminal mobility and permits place migration with no suspension of service provision (apart from temporary disconnection from the network) and with no impact on local agent execution. Several places can be grouped into a *domain* abstraction that corresponds to a network locality. In each domain, a *default place* hosts a gateway to perform inter-domain functionality.

[*] SOMA is available at `http://lia.deis.unibo.it/Research/SOMA/`.

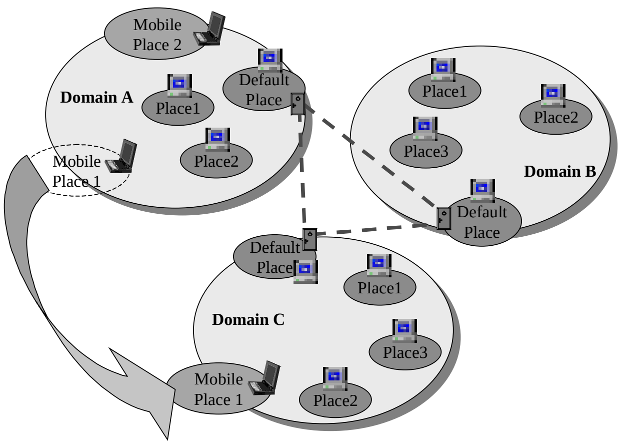


Figure 1. The SOMA locality abstractions.

In addition to these locality abstractions, the suitability of SOMA to support user and terminal mobility stems from its distributed MA-based infrastructure composed by a mobility add-on module built on top of a set of basic facilities. Figure 2 shows the SOMA layered architecture, with the basic facilities structured in two different layers. The SOMA lower layer of facilities provides:

- the *migration* facility to support the mobility of agents and resources. Resource reallocation in SOMA is achieved by encapsulating resources within agents that can move in the network either via native migration method or via CORBA Internet Inter-ORB Protocol [30] and the MASIF standard interface [16];
- the *naming* facility to associate entities with globally unique identifiers (GUID) and to organize these identifiers in name systems able to trace entities even if they move. This facility allows to put together a set of different naming systems (DNS-, CORBA-, and LDAP-compliant), characterized by different policies, and is currently implemented by a coordinated set of dedicated agents;
- the *monitoring* facility to observe the state of local resources and services and to provide this information to the application level. SOMA agents can monitor both system indicators (e.g., CPU load, file system occupation, printers status, available network bandwidth and collision rate) and application indicators (e.g., available services, program versioning, local agent states). SOMA agents exploit platform-dependent mechanisms to obtain the monitored indicators (e.g. the `psapi.dll` on Windows NT and `pstat` on Unix); they invoke this functionality provided by the hosting operating system via the JAVA Native Interface [15]. Their description is detailed in [5, 6];
- the *communication* facility to provide tools for coordination and communication between possibly mobile entities. Within the same place, agents interact by means of shared objects, such as blackboards and tuple spaces for tight cooperation. Outside the scope of the place, agents can perform coordinated tasks by exchanging messages that are delivered to agents also in case of migration;
- the *persistency* facility to temporarily suspend executing agents and to store them on a persistent medium. The facility allows agents not to waste system resources while they are waiting for external events such as the reconnection of one user or terminal where they have to yield back the results of their operations.

The SOMA upper layer of facilities provides a rich set of mechanisms and tools for interoperability and security, which we consider crucial properties to greatly widen the application opportunities of the MA technology.

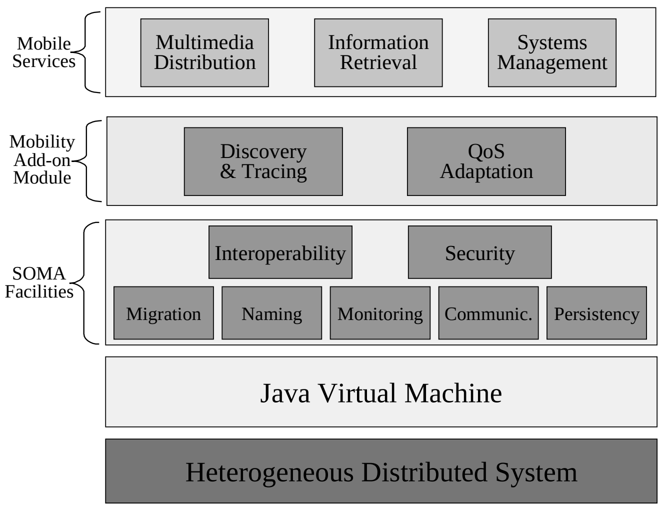


Figure 2. SOMA architecture: facilities and mobility module.

The interoperability facility allows SOMA agents to interwork with existing software and hardware components via compliance with CORBA [30]. SOMA agents can play the role of CORBA clients through either static (IDL stub) or dynamic (Dynamic Invocation Interface) invocations. In an analogous way, SOMA agents can register themselves as CORBA servers to offer access points to an application outside the SOMA system via either static IDL skeletons or dynamic skeleton interfaces. In both cases, the interoperability facility simplifies the duties of the SOMA developer by providing a set of tools to facilitate the implementation of CORBA clients/servers. Our interoperability implementation is based on the Inprise VisiBroker ORB [18]. However, it is portable, with no modification at all, on any other ORB implementation compliant to the CORBA 2.2 specification. In fact, we have only used the portable functions provided by the Internet Inter-ORB Protocol and the Portable Object Adapter [30], introduced by OMG to overcome product incompatibility. In addition, SOMA is compliant with the OMG work on the Mobile Agent System Interoperability Facility (MASIF) [16], which standardizes the basic functions an MA framework should offer for agent management and transfer to external systems, whether MA-based or not. Any external system can control agents of a MASIF-compliant MA system via the `MAFAgentSystem` interface: MASIF defines methods for suspending/resuming/terminating agents and for moving agents from one MA system to another. Agent tracking functions permit the tracing of agents registered with `MAFFinder`, introduced to provide an MA naming service, because the CORBA one is not suitable for entities that are mobile by nature. The SOMA compliance with MASIF has permitted to realize MA-based services [5] in which our mobile agents cooperate with mobile agents from Grasshopper [17], the only commercial MA platform that has already implemented the MASIF standard.

Security is a fundamental issue when dealing with mobile agents deployed in the untrusted Internet environment, where the communication network is considered insecure and any node may host the execution of possibly malicious agents. The SOMA security facility provides the basic mechanisms to ensure system wide security to mobile agents, terminals and users. In particular, SOMA supports the definition of any model of trust, to define who or what in the system is trusted, in what way, and to what extent. SOMA mobile entities are associated with certificates supported by a public key infrastructure. The SOMA security service supports the model of trust and enforces its security policies: authentication permits to identify mobile entities; authorization recognizes whether an operation is permitted on a resource; integrity guarantees that entities have not been maliciously modified during reallocation; secrecy permits to protect entities and communication messages from any exposure to malicious intrusions. SOMA makes use of X.509 certificates for authentication, and we are currently working on the integration with a commercial Public Key Infrastructure (PKI) provided by Entrust [14], to automatically distribute keys, to manage certificates and to perform all related administrative tasks. The integrity check can employ either MD5 or SHA1. Secrecy is granted when needed by the possibility of exchanging communications through a Secure Socket Layer [19]. As a final consideration, let us notice that SOMA security is provided with application-level tools, to take advantage of available standard solutions and products (e.g., the IAIK cryptographic functionality and the Entrust PKI [14, 19]). If the debate about at which level a system has to offer security is still open, the discussion concentrates on the issues of transparency, flexibility and performance [31]. Independently of the abstraction level adopted, it is important to consider security as a property to be integrated at any system layer. Only this pervasive approach, followed in the SOMA design, can achieve the full level of security, higher than the minimal one obtained by systems that add an a posteriori security strategy.

Further details about the SOMA programming framework and its implementation are presented elsewhere [4, 5] and are out of the scope of this paper, where we specifically focus on the SOMA mobility add-on module.

4 THE SOMA ADD-ON MODULE TO SUPPORT MOBILITY

The support to user and terminal mobility is based on the mobility add-on module that consists of two main components: the tracing and discovery service, and the QoS adaptation one.

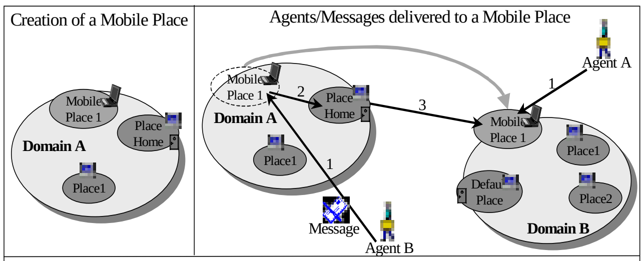
4.1 Tracing and Discovery of Mobile Entities

There are two common approaches to implement a name service that permits to trace mobile entities. In a centralized approach, a global directory service maintains the allocation of any mobile entity and it is involved in any migration [32]; in a more distributed solution, any mobile entity is associated with a corresponding care-of fixed point that keeps track of the movements [20, 28]. SOMA makes use of both approaches.

The distributed approach is followed to keep track of mobile places and users. Any mobile entity has a corresponding *home*, that is the place where the entity has been first created, and that maintains the information about the entity current position. SOMA also provides non-traceable entities not to pay the cost of updating their home at any migration. Any mobile place has a *place home*, that is the default place of the domain where it has been created, automatically notified whenever the mobile place changes its connected/disconnected status and its point of attachment. Any mobile user has a *user home*, the place of the first user registration, that keeps the user Virtual Home Environment (VHE) information [25]. Mobile agents, places and users in SOMA have GUID consisting of the identifier of the corresponding home associated with a progressive number unique in this locality. For instance, a mobile place owns a GUID of the form (*DomainID, progNumber*) where DomainID is the IP address of its *place home*. This permits to identify easily the *place home*, and to forward messages/agents for that mobile place via the corresponding home, without querying and registering with a centralized name service. Home entities represent a convenient choice for agents, terminals and users that change often location.

An additional service for mobile entities is the SOMA directory service, called *Resource Discovery* (RD). The RD service, located in each default place at a predefined port, maintains information about all the resources currently available within that domain. Resources are classified on the basis of recognized types, e.g., printers, disks, ftp servers. The service collects a list of parameters for registered resources, including both static properties (e.g., print speed, disk total space) and dynamic ones (e.g., average bandwidth measured in ftp transfer), by interacting with the monitoring basic service. When a nomadic user/terminal moves from a domain A to a new domain B, she/it can interrogate the local RD about resource availability in that domain. The nomadic user/terminal can decide either to maintain its previous bindings to its remote resources in domain A, or can ask RD to automatically re-qualify these bindings to the corresponding resources local to domain B. If the domain B does not offer the requested resource type, RD maintains the remote references. In this way, it is possible to take advantage of resource locality to reduce network traffic and to improve performance.

The SOMA naming infrastructure has demonstrated to scale well: the current position resolution of any mobile entity is completely distributed among the corresponding home entities. The RD service fits well with the SOMA hierarchical organization: each RD server has the duty of handling only local resources, i.e. the ones registered in its domain.



Agent A owns the update position of the mobile place and immediately reaches it.

Agent B and **Message** try to reach the mobile place that has moved - tunneling via the *place home*.

Figure 3. Agent/message delivery in terminal mobility.

4.2 The Support for QoS Adaptation

QoS issues are not specific for mobile computing but are particularly critical when dealing with terminal and user mobility, because of the specific characteristics of low availability and abrupt changes in network bandwidth. Two main directions are emerging in the QoS research area. Some work has defined and standardized new protocols to reserve network resources, to ensure their availability to the requesting services. In the following, we will indicate this approach as *hard QoS* [33, 9]. Other work is instead at the application level, and proposes infrastructures that enhance service provision with the possibility to require a specified QoS, to try to satisfy these requirements and to notify the service infrastructure in case of change in the currently available quality. All is done at the application level, with no strict guarantee of requirement satisfaction, but with no need to modify the underlying network layer, thus preserving the best-effort approach of the world-wide diffused Internet protocol. We define this approach *soft QoS*.

Hard QoS is the only way to ensure the real reservation and consequently the availability of the needed amount of network resources. However, the process of acceptance and deployment of new standards for network-layer protocols has demonstrated to be long and difficult, due to the large base of non-programmable and already installed network equipment in global systems such as the Internet. For instance, notwithstanding its standardization and the acceptance of its importance in the academic world, RSVP is not yet fully supported by the largest part of network routers that currently operate in the Internet. While the provision of *hard QoS* guarantees is the objective of many service architectures in fixed computing systems, it is recognized that the *soft QoS* approach is extremely suitable for mobile computing. In fact, mobile services are particularly sensible to the possibly abrupt changes in QoS depending on terminal and user mobility, and can be significantly enhanced by making them aware of the dynamically available QoS. In that way, mobile services (or the distributed infrastructure that supports them) can take application-level corrective actions to adapt themselves to the current network conditions [10]. For this reason, this paper mainly focuses on the possibility to provide an application-level QoS adaptation infrastructure for mobile services.

The SOMA QoS adaptation support exploits the monitoring functionality of the SOMA basic services. On the basis of this MA-based monitoring, SOMA implements a distributed adaptation infrastructure consisting of two types of interworking mobile agents, the Admission Controller (AC) agents and the QoS Adaptation (QoSA) ones.

AC agents are responsible for managing and controlling the local access to resources. AC agents realize an application-level reservation of the required resources by impeding the acceptation of a service request if the managed resources are not likely to satisfy the required QoS level. One AC agent is distributed on each SOMA intermediate node that actively takes part in the QoS adaptation support. Intermediate nodes can also decide not to have AC agents running on them: in this case, they behave as traditional passive nodes, and data streams generated by service components simply traverse them via tunneling. Any AC agent manages all resources local to its node and keeps track of the ones currently committed to already accepted streams. In this way, it can evaluate at any time the current resource availability and can maintain accountability information on resource consumption for any service user.

QoSA agents are the crucial components of the SOMA adaptation infrastructure and are responsible for taking corrective application-level operations in response to dynamic changes in the available QoS. Their adaptation operations (e.g., discarding, filtering, compressing, multicasting incoming streams) directly depend on the nature of the service they contribute to provide. As a consequence, they are service-specific, and any intermediate node can contemporarily host a multiplicity of different QoSA agents, typically one for each kind of service whose streams traverse the node. Therefore, while the AC agent can be supposed to be already installed at any SOMA active node, QoSA agents are necessarily distributed in the start-up phase of a new service provision. This preliminary distribution affects the starting delay of new services but does not influence performance in any successive request for service provision.

A QoSA agent registers to monitoring agents its will of being notified of variations in some specified indicators. It can specify a desired value of one system property and a threshold: when the current property value differs from the desired value more than the threshold, monitoring agents trigger the intervention of the interested QoSA. After being notified, the QoSA agent decides to operate on the service: it can simply propagate the notification to the client or to the service provider, thus delegating any decision to the final end-systems (*QoS-aware services*); it can interwork with the AC agent to request and obtain additional resources in order to maintain the initially required QoS; it can accommodate the service to the diminished resource availability by performing transformations on the exchanged data (*QoS-transparent services*), e.g., dropping frames or transcoding the format of multimedia flows.

To check the effectiveness of our QoS adaptation support, we have applied it in the field of distributed multimedia applications. The path between the video service provider and the (possibly mobile) video client is automatically determined at the moment of the user request. Monitoring, AC and QoSA agents are present on any active node of the network; this assumption is not severe because they are implemented by mobile agents that can be installed at run-time wherever they are needed. Monitoring agents observe the application-level QoS currently obtained from their local host to the next intermediate one. QoS flow specifications are expressed in terms of `<receiving-host, bandwidth, delay, loss>` tuples [12]. AC agents keep track of the already accepted streams that traverse their local node. They are in charge of answering to new connection requests from QoSA agents; they authenticate users requesting the flow and locally store information for user accounting. QoSA agents interrogate ACs: if the available resources are not enough for the required QoS, QoSAs can coordinate and reduce their resource requirements by scaling the stream. In particular, they are capable of dropping frames in Motion JPEG streams and of reducing image resolution in MPEG-2 ones [3, 13]. At the moment, we are testing our adaptation infrastructure within the framework of the MOSAICO project, by using DiVA (`http://grid.grid.unina.it/ projects/diva/`) as the service provider and the video client.

Finally, we are also working on the implementation of *hard QoS* in SOMA. We are realizing hard AC agents that are able to direct QoS requirements to the underlying network layer. At the moment, we are implementing Java interfaces to access native AAL5 services provided on ATM networks, with an approach that is analogous to the one followed in the JQoS project [21]. Work to integrate the RSVP protocol [33] into our AC agents is planned, while waiting for network equipment that supports both RSVP and the Java Virtual Machine.

5 RELATED WORK

Several research efforts are currently going on for supporting terminal mobility with a network-level approach. They are based on the introduction of network protocols that basically associate two IP addresses to any mobile host. The first address represents the current point of attachment of the mobile host to the network; the other one reflects the home address of the mobile host, i.e. the address of a care-of entity that maintains information about the mobile host current position. The various proposed network-level approaches differ in the placement of the home address functionality and in the protocol adopted to update the home address information [7]. The two most diffused proposals are *Mobile IP*, that is backward compatible with IP, but cannot achieve optimal routing (*triangle routing problem* [7]), and *IPv6*, that provides both acceptable performance and excellent scalability, but whose process of acceptance seems still long, as for any new protocol [26].

With a different perspective, the European Telecommunication Standards Institute is working on the standardization of the Universal Mobile Telecommunications System (UMTS) [25]. UMTS proposes to support personal communication directly at the level of the service infrastructure, by realizing the VHE concept. The VHE permits a mobile user to retrieve her personalized environment also when accessing services from heterogeneous and different terminals connected to heterogeneous and different networks. The VHE is the basic component of the UMTS mobility middleware, which hides the specific properties of the network from the user, and the peculiarities of user and service providers from the network.

Notwithstanding all potential advantages of adopting the MA approach, there is not much work on the use of mobile agents to support terminal and user mobility, probably due to the relative novelty of the MA technology. The MA research mainly concentrates on solving the naming issues related to terminal mobility, by providing laptops with several care-of points on the fixed network. MASE provides a mobility gateway between fixed and mobile networks [24]; Agent TCL implements the same functionality via a docking station in charge of forwarding agents and messages to the mobile terminal [23]; finally, the Discovery system permits to notify all interested agents of the dis/connection of a mobile terminal [27]. Other MA-based proposals for user mobility mainly focus on the provision of VHE and directory services [28].

In the area of QoS adaptation, first results are emerging in the realization of distributed infrastructures for supporting QoS monitoring and dynamic service adaptability. The largest part of the proposals comes from the active network area: several research activities address the possibility to reserve resources on the path between the service provider and its clients, and to dynamically inject application-specific multicast protocols [11]; others propose intermediate hosts that play the role of active filters to dynamically adapt the exchanged information to the currently available bandwidth, especially in the multimedia distribution application domain [1, 2]. However, there is not a general agreement on the fact that the network-level approach typical of active networks is the most suitable for the provision of effective, programmable and adaptive service infrastructures [3]. Among the proposals at the application-level, some work is also emerging in the field of multi-agent systems, mainly focused on establishing and maintaining as far as possible a user-defined level of QoS [29]. The multi-agent approach to QoS adaptation is similar to ours from many points of view, but multi-agent service components suffer from a lower degree of flexibility, since they cannot dynamically migrate and have to be present on the involved hosts before the starting of service provision.

6 CONCLUDING REMARKS

The paper describes the adoption of mobile code technologies to support user and terminal mobility, by describing the implementation of the two components of the SOMA mobility add-on module, the tracing and discovery service and the QoS adaptation one. The SOMA modular organization has permitted to design and implement the mobility support module incrementally as a natural extension of the already available SOMA infrastructure.

In addition, apart from feasibility, the first experience in the system usage has exhibited efficient and scalable performance for the implemented mechanisms to support mobility. In particular, the tracing and discovery service has shown good scalability while increasing the number of mobile entities involved, mainly due to the delegation and distribution of registering duties over all involved places, without forcing to maintain a centralized and strictly consistent naming service. Also the first prototypes of QoS adaptation components are demonstrating the viability of the approach, at least if considered as separated basic mechanisms. However, since the characteristics of the QoS adaptation components are tightly connected to the nature of the interested service, we expect to obtain significant feedback from the performance results we are collecting in the implementation of a multimedia distribution service capable of adapting to groups of heterogeneous QoS-requesting PDA receivers. This feedback will guide our work of refinement of the already implemented QoSA agents.

Finally, thanks to its properties of modularity and interoperability, the SOMA mobility support has demonstrated to be an open and extensible mobility framework. We are currently working on its further development to answer the specific needs of different application areas for mobility, such as network and systems management in case of nomadic administrators, and user-asynchronous information retrieval in geographically distributed heterogeneous repositories.

7 ACKNOWLEDGMENTS

Work carried out under the financial support of the Italian Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST) in the framework of the Project "MOSAICO: Design Methodologies and Tools of High Performance Systems for Distributed Applications".

8 REFERENCES

[1] Amir, E., McCanne, S., and Zhang, H. An Application-level Video Gateway. Proc. ACM Multimedia'95, Nov. 1995.

[2] Amir, E., McCanne, S., and Katz, R. Receiver-driven Bandwidth Adaptation for Light-weight Sessions. Proc. ACM Multimedia'97, Nov. 1997.

[3] Amir, E., McCanne, S., and Katz, R. An Active Service Framework and its Application to Real-Time Multimedia Transcoding. ACM SIGCOMM98, Vancouver, Sep. 1998.

[4] Bellavista, P., Corradi, A., and Stefanelli, C. A Secure and Open Mobile Agent Programming Environment. Proc. ISADS99, Tokyo, Japan, Mar. 1999.

[5] Bellavista, P., Corradi, A., and Stefanelli, C. An Open Secure Mobile Agent Framework for Systems Management. Journal of Network and Systems Management, Vol. 7, No. 3, Sep. 1999.

[6] Bellavista, P., Corradi, A., Stefanelli, C., and Tarantino, F. Mobile Agents for Web-based Systems Management. Inter-

net Research, MCB University Press, Vol. 9, No. 5, Nov. 1999.

[7] Bhagwat, P., Perkins, C., and Tripathi, S. Network Layer Mobility: An Architecture and Survey. IEEE Personal Communications, Vol. 3, No. 3, June 1996.

[8] Bolliger, J., and Gross, T. A Framework-Based Approach to the Development of Network-Aware Applications. IEEE Trans. Software Engineering, Vol. 24, No. 5, May 1998.

[9] Busse, I., Deffner, B., and Schulzrinne, H. Dynamic QoS control of multimedia applications based on RTP. Computer Communications, Vol. 19, No. 1, Jan. 1996.

[10] Chalmers, D., and Sloman, M. A Survey of Quality of Service in Mobile Computing Environments. IEEE Communications Surveys, http://www.comsoc.org/pubs/surveys, 2nd Qrt. 1999.

[11] Chawathe, Y., Fink, S., McCanne, S., and Brewer, E. A Proxy Architecture for Reliable Multicast in Heterogeneous Environments. Proc. ACM Multimedia'98, Sep. 1998.

[12] Couloris, G., Dollimore, J., and Kindberg, T. Distributed Systems: Concepts and Design. Addison-Wesley, 1994.

[13] Delgrossi, L., et al. Media Scaling in a Multimedia Communication System. ACM Multimedia Systems, Vol. 2, 1994.

[14] Entrust Technologies. Entrust. http://www.entrust.com/.

[15] Flanagan, D. Java Power Reference. O'Reilly & Associates, March 1999.

[16] GMD FOKUS, and IBM Corp. Mobile Agent Facility Specification. Joint Submission supported by Crystaliz Inc., General Magic Inc., the Open Group, OMG TC Document orbos/97-10-05, ftp://ftp.omg.org/pub/docs/orbos/, 1998.

[17] IKV++ GmbH. Grasshopper. http://www.ikv.de/products/grasshopper/.

[18] Inprise. VisiBroker for Java. http://www.borland.com/ visibroker.

[19] Institute for Applied Information Processing and Communications. IAIK JCE. http://jcewww.iaik.tu-graz.ac.at/.

[20] Ioannidis, J., Duchamp, D., and Maguire, G. IP-Based Protocols for Mobile Internetworking. ACM SIGCOMM Computer Communication Review, Vol. 21, No. 4, Sep. 1991.

[21] Kassler, A., Christein, H., and Schulthess, P. A Generic API for Quality of Service Networking based on Java. Proc. ICC'99, Vancouver, Canada, June 1999.

[22] Kim, P.J., and Yoon, S.H. Mobile Agent System Architecture for Mobile Computing by Using Proxy Technology. Int. Conf. On Telecommunications, Greece, 1998.

[23] Kotz, D., et al. Agent TCL: Targeting the Needs of Mobile Computers, IEEE Internet Computing, Vol. 1, No. 4, 1997.

[24] Kovacs, E., Rohrle, K., and Reich, M. Integrating Mobile Agents into the Mobile Middleware. Proc. Mobile Agents International Workshop (MA'98), Springer-Verlag, pp. 124-135, Germany, 1998.

[25] Kreller, B., et al. UMTS: A Middleware Architecture and Mobile API Approach. IEEE Personal Communications, Vol. 5, No. 2, Apr. 1998.

[26] Kumar, A. Third Generation Personal Communication Systems. IEEE Int. Conf. on Personal Wireless Communications, pp. 313-318, New York, 1996.

[27] Lazar, S., Weerakoon, I., and Sidhu, D. A Scalable Location Tracking and Message Delivery Scheme for Mobile Agents. IEEE Int. Workshop on Enabling Technologies, 1998.

[28] Lipperts, S., and Park, A. An Agent-based Middleware: a Solution for Terminal and User Mobility. Int. Journal of Computer and Telecommunication Networking, 1999.

[29] Nait-Abdesselam, F., Agoulmine, N., and Kasiolas, A. Agent-based Approach for QoS Adaptation in Distributed Multimedia Applications over ATM. Int. Conf. on ATM, June 1998.

[30] Object Management Group. CORBA/IIOP Rev 2.3. OMG Document formal/98-12-01, http://www.omg.org/library/, Dec. 1998.

[31] Oppliger, R. Security at the Internet Layer. IEEE Computer, Vol. 31, No. 9, Sep. 1998.

[32] Perkins, C., and Harjono, H. Resource Discovery Protocol for Mobile Computing. IFIP World Conf. on Mobile Communications, pp. 219-236, London, 1996.

[33] Zhang, L., Deering, S., Estrin, D., Shenker, S., and Zappala, D. RSVP: a new resource ReSerVation Protocol. IEEE Network, Vol. 7, No. 5, Sep. 1993.