# Mobile Agent Middleware for Mobile Computing

**Mobile agent-based middleware shows promise for providing an advanced infrastructure that integrates support protocols, mechanisms, and tools to permit communication and coordination of mobile entities.**

*Paolo Bellavista*
*Antonio Corradi*
University of Bologna

*Cesare Stefanelli*
University of Ferrara

The integration of the Internet with telecommunication networks promises a distributed computing infrastructure that provides globally available services. The Internet's diffusion permits an almost ubiquitous availability of attachment points, allowing users access to its information services irrespective of their location. In addition, advances in cellular telecommunications and device miniaturization let increasing numbers of portable devices connect to the Internet.[1]

Several forms of mobility emerge in this scenario. *User mobility* requires providing users with a uniform view of their preferred working environments—user preferences and subscribed services—independent of their current positions in the network.[2] *Terminal mobility* allows devices to transparently move and connect to different points of attachment. *Mobile access* is an emerging issue that involves the dynamic adaptation of mobile-aware resources and services that mobile users and terminals can automatically retrieve regardless of their current location.[3]

Mobile computing requires an advanced infrastructure that integrates suitable support protocols, mechanisms, and tools. This mobility middleware should dynamically reallocate and trace mobile users and terminals and permit communication and coordination of mobile entities. In addition, open and untrusted environments must overcome system heterogeneity and grant the appropriate security level. Solutions to these issues require compliance with standards to interoperate with different systems and legacy components and a reliable security infrastructure based on standard cryptographic mechanisms and tools.

Many proposals suggest using mobile agent technology middleware to address these issues.[4-6] A mobile agent (MA) moves entities in execution together with code and achieved state, making it possible to upgrade distributed computing environments without suspending service.

We propose three mobile computing services: user virtual environment (UVE), mobile virtual terminal (MVT), and virtual resource management (VRM). UVE provides users with a uniform view of their working environments independent of current locations and specific terminals. MVT extends traditional terminal mobility by preserving the terminal execution state for restoration at new locations, including active processes and subscribed services. VRM permits mobile users and terminals to maintain access to resources and services by automatically requalifying the bindings and moving specific resources or services to permit load balancing and replication.

## MOBILE AGENTS IN MOBILE COMPUTING

Mobile computing benefits from the asynchronicity between user requests and terminal operations and their execution. For example, wireless connections impose strict constraints on available bandwidth and communication reliability minimizing connection time for wireless device support. The MA paradigm does not need continuous network connectivity because connections last only long enough to inject agents from mobile terminals into the fixed network. With autonomous agents, users can access services even if the terminal disconnects because the agents deliver the results upon reconnection.[4-6]

Mobility middleware's location awareness facilitates service-specific optimization and allows users to adapt to

0018-9162/01/$10.00 © 2001 IEEE

local resources. Mobile users can change location and dynamically tailor mobility-enabled applications to the properties and characteristics of their network connections and hardware devices. Location awareness facilitates allocation visibility up to the application level supporting dynamic quality of service (QoS) adaptation to local needs.[5,6] In addition, MA simplifies dynamic personalization by following user movements and tailoring service depending on personal preferences.

Mobility stresses the importance of flexible and extensible middleware. The dynamic distribution of code typical of MA platforms requires new components and protocols to adapt to evolving service and user requirements. Mobile agents provide flexibility by moving code and preserving the state the computation produces.

Mobility raises significant security issues for authentication of mobile users and terminals, authorization to access system resources, and communications' secrecy and integrity assurance. After the pioneering work of IBM Aglets,[7] recent MA platforms provide flexible mechanisms and policies to grant the most suitable security level.[8,9] For example, many MA systems integrate with public key infrastructures, simplifying authentication of mobile users and terminals. The "Approaches and Solutions to Mobile Computing" sidebar provides a summary of the findings in recent studies investigating mobility issues.

## Approaches and Solutions to Mobile Computing

Current research focuses on three aspects of deploying services in a mobile computing environment: user mobility, terminal mobility, and mobile access to resources.

### User Mobility

Facilitating user mobility independent of both terminal properties and current physical location requires an infrastructure that authenticates user access and organizes the working environment according to information in the user profile. The profile includes the graphical interface information and all user preferences such as the default language, required security level, and subscribed services. Profiles can also include user-specified information to adapt the working environment to the current terminal's hardware characteristics. For example, if the user connects via a PDA with limited bandwidth and limited graphic resolution, discarding large images makes sense.

The Universal Mobile Telecommunications System proposes a service infrastructure based on the concept of a virtual home environment[1] in which users have access to the same personalized features, interfaces, and services regardless of the current hosting network. Profile preferences, terminal equipment, and current network conditions determine the user's working environment. The World Wide Web Consortium promotes the composite capability/preference profile, a standard proposal for the representation of profile information and the exchange protocol, based on the resource description format encoded in XML.[2] Developers are considering CC/PP to tailor the provision of Internet services to the specific characteristics of WAP mobile phones.[3] In addition, the Foundation for Intelligent and Physical Agents is working to define an agent interoperability framework for nomadic support that provides information for user profile management and mobile device characteristics.[4]

Several proposals integrate both user preferences and information about current terminal characteristics in the user profile. Although personalization and adaptation of services both need user- and terminal-dependent information, the two dimensions should be cleanly decoupled. User profiles should contain only user-related information, and devices should independently inform the support infrastructure of their characteristics. This approach achieves maximum flexibility and reusability so that the same user can exploit a set of different terminals with different characteristics.

### Terminal Mobility

Many state-of-the-art proposals address terminal mobility at the lower layers of the OSI protocol stack. Network layer protocols such as mobile IP[5] associate a mobile host with two IP addresses. The first address represents the current point of attachment to the network, while the second reflects the mobile host's home address—the address of a fixed care-of entity that traces the mobile host's current position. Mobile IP is backward compatible with IP but cannot achieve optimal routing because it always requires packets to pass through the care-of entity. Another solution, IPv6,[5] adopts an approach similar to mobile IP but also provides acceptable performance and excellent scalability by permitting senders to cache information about the current location of their mobile destinations. As with all new protocol proposals, however, accepting and adopting IPv6 is likely to be a long process.

The IEEE 802.11 standard addresses the issue of wireless communication in local area networks. This standard covers a broad area ranging from the physical-media layer that defines frequencies and their usage to the media-access layer that defines basic packet framing and headers. Unfortunately, not all wireless device producers have accepted 802.11, which makes full multivendor interoperability a long-term hope at best.[6]

The lengthy acceptance process new protocols undergo has motivated proposals for programmable network architectures that simplify protocol prototyping and deployment.[7] While programmable networks are still an open research issue, other TCP/IP-based solutions for specific aspects of mobility support exist.

Interoperability allows mobile users and terminals to interact with available resources and services when moving between hosting environments. To address similar problems, MA research promotes interoperable and standard interfaces. For example, some MA platforms already are compliant with Corba and related standards, such as the Mobile Agent Systems Interoperability Facility and the Foundation for Intelligent Physical Agents specifications.[10] MASIF defines standard interfaces for the basic functions of agent management and transfer between heterogeneous MA systems. FIPA standardizes the general architecture of agent platforms and focuses on interoperable agent communication languages.

Some MA environments support various forms of mobility. MA research mainly focuses on terminal mobility. Using a mobile application support environment, the ACTS OnTheMove project adapted an existing MA system to provide a gateway for mobility between fixed and wireless networks.[4] Dartmouth Agent TCL writes agents in different languages—such as TCL, Java, and Scheme—and implements a docking station in charge of forwarding agents and messages to mobile terminals.[5] The Discovery MA system supports terminal mobility by implementing an infrastructure that notifies all interested agents of distributed events, such as the connection and disconnection of a mobile device.[11] Other MA proposals concentrate

For example, the Dynamic Host Configuration Protocol can automate the configuration of nomadic hosts by dynamically assigning temporary IP addresses.[8]

Network-layer approaches may not provide a flexible enough solution to fundamental mobility issues such as security and interoperability that appear at a higher level of abstraction and can benefit from standard tools at the application level.[9]

### Mobile Access to Resources

Research activities have not fully addressed the issue of maintaining access to available resources and services while moving. This capability requires mobility-enabled naming solutions to maintain the information about availability and allocation of resources and services. The two main categories of naming solutions suitable to mobility applications are discovery and directory services.

Discovery services usually employ simple protocols to obtain information about entity location, such as address and simple configuration data, with a minimal knowledge of hosting environments. Recent research has produced widely accepted distributed protocols to maintain information about current resource availability in local networks and to answer simple queries. Different implementations reflect the different types of resources they classify, from simple embedded devices, as in Microsoft's simple service discovery protocol for its proprietary Universal Plug and Play,[10] to more complex service components, such as those in Jini and the Service Location Protocol.[8]

Directory services usually organize names and properties for registered entities flexibly and facilitate browsing all registered information with complex search patterns. Apart from the traditional work on directory standards, such as the X.500 directory access protocol, the Internet community has defined the simplified Lightweight Directory Access Protocol that runs directly on top of TCP/IP.

Using discovery and directory services to automatically maintain and update bindings to resources and services in mobility scenarios is still in its infancy. The first proposals agree on the necessity of using middleware to transparently reestablish TCP/IP connections in nomadic computing[11] and the need for automatic adaptation of service flows to the type of resources currently available to PDAs.[12]

### References


1. European Telecommunications Standards Institute, *Universal Mobile Telecommunications Systems*; http://www.etsi.org/umts/.
2. W3 Consortium, "Composite Capability/Preference Profiles (CC/PP)"; http://www.w3.org/TR/NOTE-CCPP/.
3. Wireless Application Protocol (WAP) Forum; http://www.wapforum.org/.
4. Foundation of Intelligent Physical Agents (FIPA); http://www.fipa.org/.
5. P. Bhagwat, C. Perkins, and S. Tripathi, "Network Layer Mobility: An Architecture and Survey," *IEEE Personal Comm.*, June 1996, pp. 54-64.
6. Wireless Local Area Networks Committee, "IEEE P802.11"; http://grouper.ieee.org/groups/802/11/index.html.
7. K. Psounis, "Active Networks: Applications, Security, Safety, and Architectures," *IEEE Comm. Surveys*; http://www.comsoc.org/pubs/surveys, 1999.
8. C. Perkins, "Plug & Play Internet," *Internet Computing*, July/Aug. 1999, pp. 42-44.
9. J. Bolliger, and T. Gross, "A Framework-Based Approach to the Development of Network-Aware Applications," *IEEE Trans. Software Eng.*, May 1998, pp. 376-390.
10. Microsoft Corp., "Universal Plug and Play Forum Resources"; http://www.upnp.org/resources.htm.
11. J.S. Hansen et al., "Dynamic Adaptation of Network Connections in Mobile Environments," *Internet Computing*, Jan./Feb. 1998, pp. 39-48.
12. A. Fox et al., "Adapting to Network and Client Variation Using Infrastructural Proxies: Lessons and Perspectives," *IEEE Personal Comm.*, Oct. 1998, pp. 8-24.

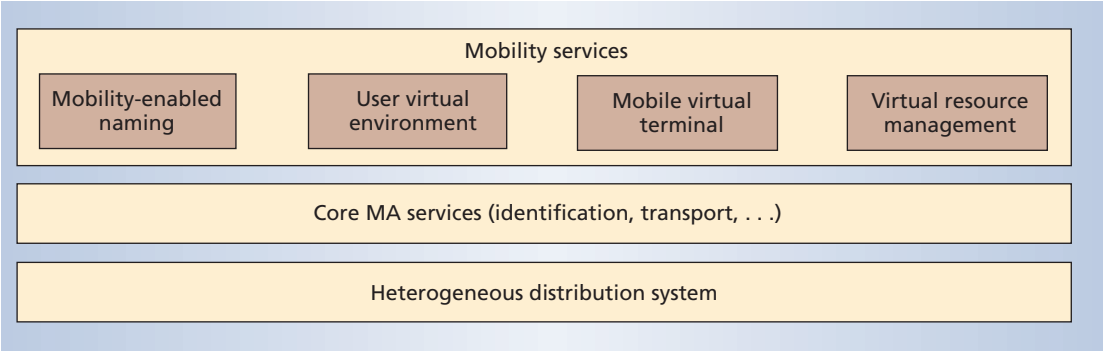



*Figure 1. MA-based middleware. The architecture is organized in layered services. The core services layer is refined by the addition of a mobility services layer that supports user mobility, terminal mobility, and mobile access to resources.*

on user profiling, VHE, and directory services.[6] How to maintain and requalify the bindings of mobile entities with (possibly mobile) resource and service components remains an open research area.

## MA-BASED MOBILITY MIDDLEWARE

Implementing MA-based middleware to support mobile computing requires extending the MA platform architecture, which is generally organized in layered services.[8,9] In particular, MA implementation entails adding a new layer of mobility services to refine the usually provided core services. As Figure 1 shows, this layer includes a mobility-enabled naming service and three services—user virtual environment, mobile virtual terminal, and virtual resource management—to support user mobility, terminal mobility, and mobile access to resources.

### Mobility naming services

A mobility-enabled naming service capable of tracing entities that move in the global Internet environment requires basic mechanisms to assign globally unique identifiers (Guids). For mobile purposes, a single centralized identification authority would introduce unacceptable reliability and overload problems. Frequently, proposed solutions partition the global environment in non-overlapping regions, with each regional identification authority in charge of serving its zone. Guids are usually very low-level identifiers; naming services can associate entities with several names that are Guid aliases, suitable for service developers and final users.

Naming services not only translate a high-level name into the corresponding Guid, but also maintain information about current entity location. Consequently, mobility stresses the naming service to the limit because it should intervene at any migration and at any mobile entity search. Different naming services—discovery and directory—can support mobility, and applications should choose the most suitable service depending on their specific domain. The discovery service generally provides information to its clients on a local scale without requiring specific knowledge; a client typically requests the service with a broadcast in the local network. Independent distributed servers that provide information for a specific locality usually implement discovery services. Because of restricted visibility scope and limited query flexibility, many discovery services exhibit excellent performance.

The directory service gives global visibility to all authorized clients. It permits entities to register in flexible hierarchies and can answer advanced queries with pattern matching on complex attributes. Distributed servers implement scalable directory services: they manage partially replicated copies of the names space and coordinate with each other to answer global requests. Caching copies of frequently asked information can improve performance.

Discovery and directory services differ in visibility scope—local versus global; flexibility—rigidly predefined and simple structure versus flexible content and organization; and performance—limited low-level efficient protocols versus complete but expensive high-level searching and registering operations. All these properties are relevant for mobile computing and suggest that a naming service should integrate the different solutions. Because of the overhead of registering operations, only globally available entities that do not move too often should take advantage of directory services. Discovery solutions, on the other hand, provide access to entities that move often within the same locality. Both naming services require security solutions to restrict access only to authorized users.

### User virtual environment

The UVE service lets users connect to the Internet at different locations, possibly via heterogeneous terminals, while maintaining the personal configurations indicated in their user profiles. Users specify profile information at the first registration, which they can modify at any time. This information includes common attributes such as the preferred icon arrangement on the display, as well as more complex data such as personal X.509 certificates, the resources requested to the hosting environment for ordinary tasks, and user constraints to direct QoS adaptation to the current terminal type. For example, in a connection established via mobile phone, the user can direct the mobil-

ity middleware to discard large attachments from incoming mail.

UVE stores user profiles at the "user home"—a fixed host where the user first registered. For increased reliability and efficiency, UVE replicates and caches user profile copies at several other locations. UVE makes user profiles globally available by exploiting a directory service that can transparently map requests to the most convenient UVE server available, according to any user-preferred metric such as network distance, response time, load balancing, and so on.

UVE benefits greatly from an MA-based implementation because mobile agents simplify dynamic distribution of UVE information. In addition, MA support provides automatic and manual mechanisms that save the user session's state and then transports and restores those settings to a new location where the user can find previously configured services, possibly adapted and scaled. UVE also yields execution results to users regardless of their current location. When a user is disconnected, UVE commands the MA-based middleware to temporarily freeze the agents with the results. These agents then restart only at user reconnection.

UVE must ensure privacy of user profile information. Towards this end, the security service must enforce user authentication and support a secure communication channel. There are several cryptographic protocols and infrastructures that offer standard solutions suitable for the Internet environment, for example, secure socket layer and public key infrastructures.

Add-ons can enhance the UVE service. For a regularly mobile user, the MA infrastructure can arrange result delivery in advance. If the user expresses repeated interest in a particular query's results, such as those for selected stock updates, the MA can send the results to locations that the user specifies.

## Mobile virtual terminal

MVT supports the migration of mobile devices among different locations by permitting the mobile terminal to continue execution while preserving the state of its interactions with the network. The protocols cited in the sidebar provide the first steps toward MVT support, but they address only network connectivity. MA-based mobility infrastructures, instead, simplify the tracing of mobile terminals, dynamic rebinding of resources and services, support of out-of-band computations, and persistence of the interaction state.

MA platforms facilitate the tracing of mobile terminals. We generally use care-of solutions, discovery services, and directory services to establish traceability after migration. An agent at a fixed location can act as the care-of entity, forwarding messages for its mobile device. A discovery service might keep track of a mobile terminal within a network locality. A directory service makes mobile devices visible to all authorized entities in the global system but imposes a larger run-time overhead. An integrated middleware architecture must provide service developers with support for any solution.

In addition, any mobile terminal should continue to access the needed network resources and services independently of its location. MVT interacts with VRM to provide several solutions. MVT requalifies terminal references by binding to equivalent resources and services in the new locality. If requalification is impossible or undesired, MVT maintains references to currently remote resources. It also supports the creation of new bindings to previously unknown resources and services.

**MA platforms provide a migration service that serializes both agent code and execution state into streams suitable for network transfer and storage persistency.**

The MA-based MVT implementation supports out-of-band computations—noninteractive operations performed while the terminal is disconnected. Before terminal disconnection, MVT commands agents running on the mobile device to migrate to hosts in the fixed network. Agents operate asynchronously with the disconnected terminal. When a mobile terminal reconnects at another attachment point, MVT delivers waiting agents and messages to that device.

In addition, MA platforms provide a migration service that serializes both agent code and execution state into streams suitable for network transfer and storage persistency. MVT exploits MA serialization to marshal and unmarshal the terminal session state on stable storage media and continue the execution from recovered information. This persistency checkpoint is helpful in critical situations like a connectivity loss or a power shortage.

MVT addresses the security and interoperability issues also inherent in MA-based implementation. MVT exploits MA security tools—authentication via X.509 certificates, flexible authorization policies, and standard cryptographic libraries—to grant security to mobile terminals. It employs the interoperability solutions available in MA platforms—Internet protocols adoption and Corba compliance—to integrate mobile terminals with heterogeneous resources and services.

## Virtual resource management

VRM maintains information about the properties and current location of available resources and services. For terminal mobility, VRM implements the server-side functions to establish dynamic connections between mobile terminals and needed resources, just as MVT provides the client-side functions.

An MA-based implementation of VRM facilitates the modification and migration of system resources and services at runtime, enabling complex manage-

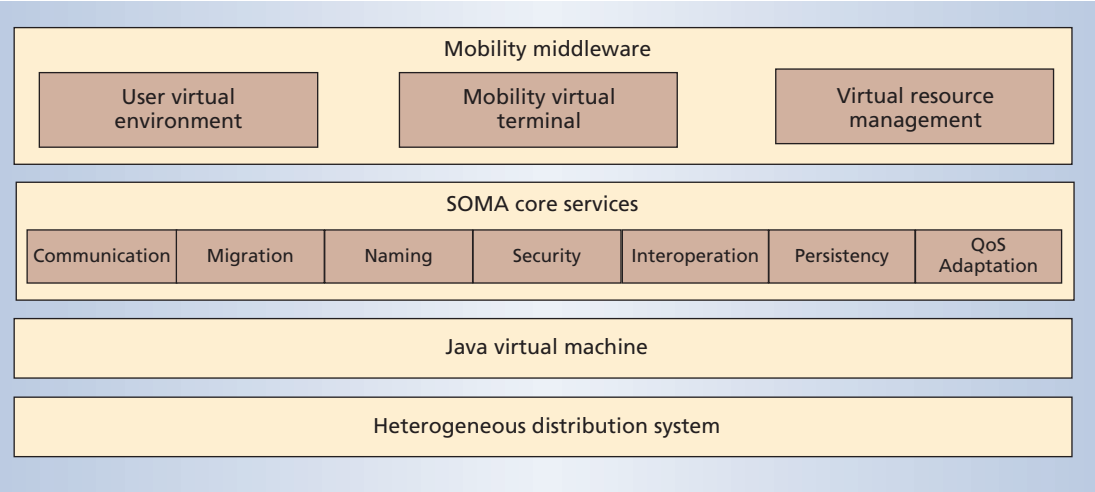



*Figure 2. SOMA architecture. The layered infrastructure includes mobility middleware, core services, a Java virtual machine, and a heterogeneous distributed system for designing, implementing, and deploying MA-based Internet applications.*

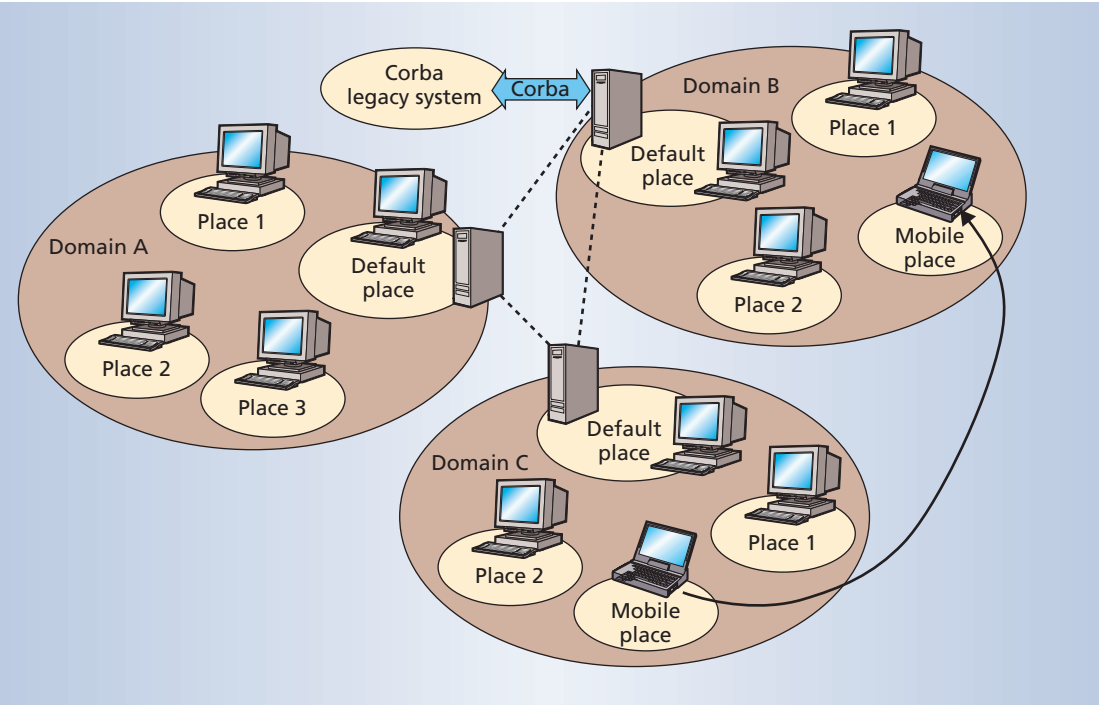



*Figure 3. SOMA locality abstractions. Each node provides at least one place for agent execution, and places are grouped into domains. Each domain has a default place in charge of interdomain routing and Corba-based interoperability.*

ment operations. For example, an administrator can configure VRM to favor locality in resource access and to balance the system load by dynamically redistributing components. Moreover, MA-based middleware can maintain pending bindings to agent-wrapped migrated resources.

MA's flexible naming service is beneficial for mobile resources and services. VRM employs discovery and directory solutions depending on resource and service requirements. For example, VRM can direct the discovery service to retrieve a folder for one developer team working locally on a LAN. If the project requires the collaboration of other departments, VRM can register the resource at the directory service to provide wider accessibility to all authorized developers.

Global scenarios require VRM to address resource and service heterogeneity. MA systems employ Java object technology to wrap resources into agents that standardize interfaces and control access; MA frameworks also use Corba to simplify integration with legacy systems.[8,9] Agent wrapping makes it possible to use established MA security mechanisms and policies to control, monitor, and log accesses to resources. Migrating mobile agents during network transmission and hosted execution requires additional measures to ensure the security of resources and service components.

The MA technology has already faced similar problems to ensure the integrity and privacy of mobile agents during both network transmission and hosted execution.

## SOMA-BASED MIDDLEWARE

The secure and open mobile agent (SOMA) distributed programming framework[9] is a Java-based platform that provides a layered service infrastructure for designing, implementing, and deploying MA-based Internet applications. As Figure 2 shows, SOMA's architecture consists of four layers. The mobility middleware layer implements UVE, MVT, and VRM services. The core services layer includes communication, migration, naming, security, interoperability, persistency, and QoS adaptation services. The architecture's other two layers are a Java virtual machine and a heterogeneous distributed system.

SOMA offers locality abstractions to describe any kind of interconnected system, from simple intranet LANs to the Internet. Each node provides at least one environment for agent execution called *place*—an agent execution environment. SOMA groups several places into domain abstractions that correspond to network localities. In each domain, a default place controls interdomain routing and integration with legacy components via Corba. In the example shown in Figure 3, a mobile place has migrated from domain C to domain B. The mobile place enhances the place locality abstraction with specific functions for automatic reconfiguration when changing domains.

### SOMA mobility services

SOMA's persistency service lets application designers and system administrators suspend agent execution by storing the agent's state on disk. The persistency service uses SOMA's migration service to serialize both agent code and data, saving the information in persistent storage. Persistency minimizes the consumption of system resources while agents wait for a disconnected resource. In addition, persistency provides fault-tolerance by duplicating and storing agent copies before starting critical operations. MVT employs persistency to freeze and wake up agents and messages when user and terminal disconnects and reconnects occur.

SOMA naming derives from care-of mechanisms for locating mobile agents and places. The care-of for any mobile agent to be traced should be located where it was first created (agent home). Similarly, the care-of for any mobile place is located at the instantiation domain's default place (place home). SOMA's middleware transparently updates agent homes at their migration and places homes at their connection or disconnection. SOMA mobile agents and places have Guids independent of their current position. Guids consist of the corresponding home's identifier associated with a number unique in the home locality. For example, a mobile place owns a Guid of the form *DomainID, progNumber* where *DomainID* is the address of its place home. This solution permits immediate identification of the home, without querying the naming service.[9] In addition to providing basic naming mechanisms, SOMA middleware integrates a discovery protocol and an LDAP-based directory service.

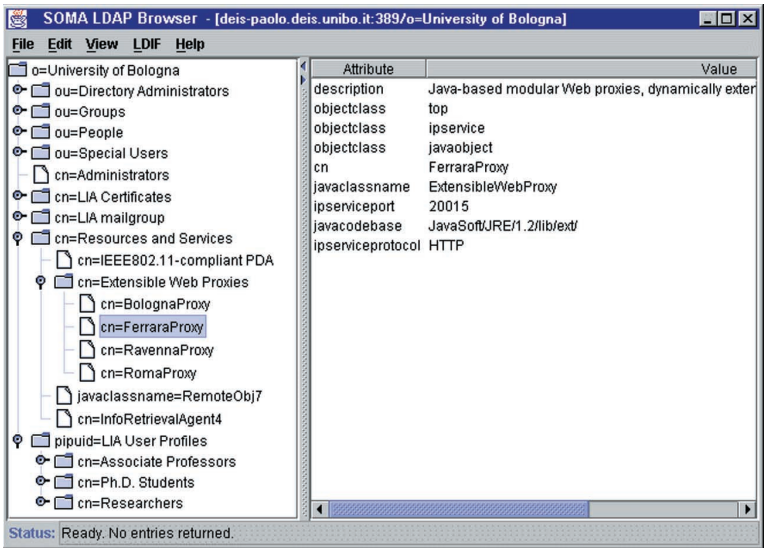


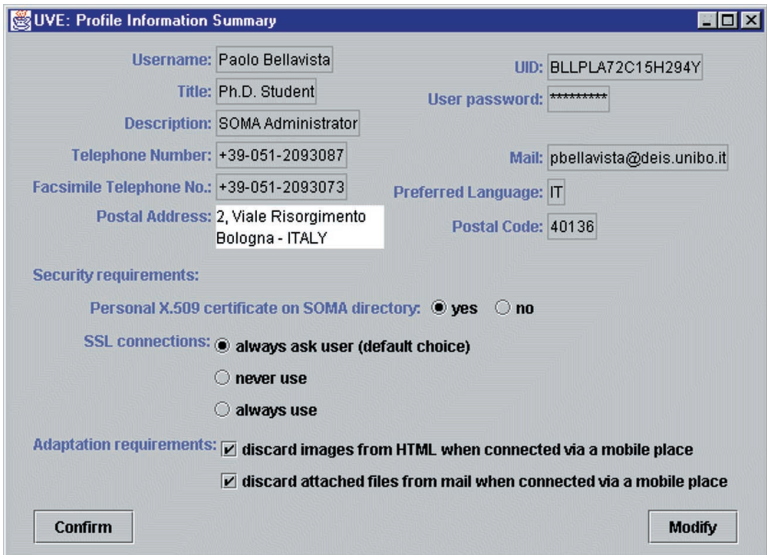

***Figure 4. SOMA's LDAP directory service as applied in a university computer science department. The SOMA-based UVE exploits the directory service to retrieve the LIA user profiles independently of the user's current point of attachment.***

The discovery service provides the default solution for resource naming within a SOMA domain. A broadcast protocol registers and deregisters resources at the discovery server located at the default place. The expected low frequency of resource migration and locality resource access suggest the choice of discovery. To date, SOMA has used a proprietary discovery protocol, but we are implementing an SLP-compliant solution.[8]

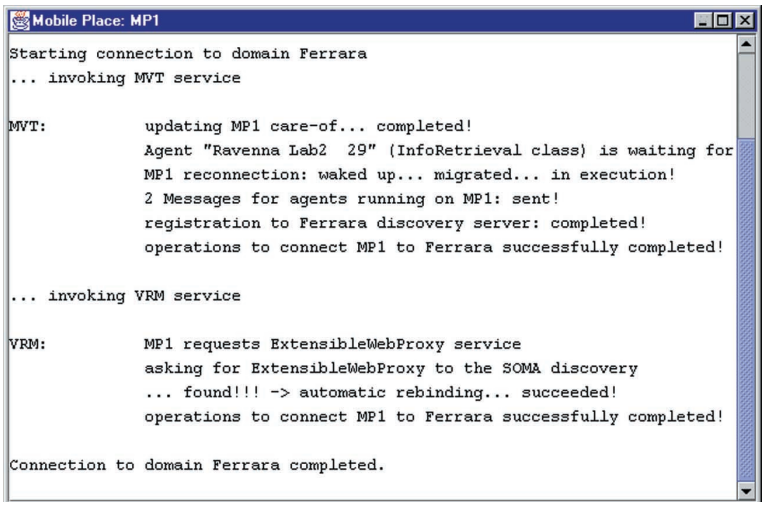
All entities that need global visibility register to the LDAP-based SOMA directory service. The LDAP directory server keeps its entity names and coordinates with other servers to maintain global consistency and to resolve external names. For example, in our department, the LIA user profiles shown in Figure 4 are registered at the directory service to provide mobile users with corresponding preferences from any location. After migration, resources can override the default discovery solution and register with the SOMA directory for wider accessibility.

### SOMA mobility usage scenario

Figure 5a shows the UVE interface for profile modification. The profile reports personal data, user security requirements, user suggestions for the SOMA QoS




(a)




(b)

***Figure 5. SOMA usage scenario. (a) UVE user profile information summary. (b) Mobile place interactions with the MVT and VRM services at reconnection in the Ferrara domain. Using SOMA mobile agents to implement UVE, MVT, and VRM services facilitates mobility middleware distribution and simplifies enforcement of balancing and replication policies.***

adaptation service, and the availability of X.509 certificates in the SOMA directory.

In the example in Figure 5b, the MP1 mobile place is coming from the Bologna domain and requesting entrance to the new hosting domain at Ferrara. MVT updates the care-of at the MP1 home with Ferrara's current location information. One mobile agent and two messages sent to agents in execution are frozen at the MP1 home during MP1 disconnection. MVT forwards all suspended entities to MP1 in Ferrara. Then, VRM provides for requalification of Web access according to the user profile.

Before leaving Bologna, the user owned a binding to the corresponding Web proxy server in the Bologna domain. At reconnection in Ferrara, VRM first interrogates the discovery service about the availability of one equivalent Web proxy. If a local proxy is available, VRM rebinds MP1 to it; otherwise, VRM asks the directory. If many functionally equivalent Web proxies were available, as in Figure 3, MVT would have prompted the user for a choice. Finally, MVT updates the Ferrara discovery server to include MP1, which can work as a SOMA fixed place.

SOMA mobile agents implement the UVE, MVT, and VRM services. This facilitates distribution of the mobility middleware and simplifies enforcement of balancing and replication policies. In addition, the Java-based implementation of the mobility middleware overcomes platform heterogeneity and applies to the open Internet.

Mobile-code technologies already address the problems that stem from changing the allocation of executing entities. Guidelines drawn from these solutions help identify and fulfill mobile computing requirements. Available mobile-code technologies only partially address the problems raised by changing the allocation of executing entities. The implementation of our mobility middleware confirmed that a layered and modular MA-based service infrastructure can support a wide range of mobile computing requirements. SOMA's UVE, MVT, and VRM service layer provides a coordinated and flexible middleware that application designers can use to design and deploy Internet services that operate in an environment where users, terminals, resources, and services are all mobile. ✸


## Acknowledgments

This research was supported by the Italian National Research Council through the Global Applications in the Internet Area and by the Italian Ministry of University in Infrastructure for QoS in Web Multimedia Services with Heterogeneous Access.


## References


1. T. Lewis, "Information Appliances: Gadget Netopia," *Computer*, Jan. 1998, pp. 59-68.
2. A. Kumar, "Third-Generation Personal Communication Systems," *Proc. IEEE Int'l Conf. Personal Wireless Comm.*, IEEE Press, Piscataway, N.J., 1996, pp. 313-318.
3. J. Jing, A.S. Helal, and A. Elmagarmid, "Client-Server Computing in Mobile Environments," *ACM Computing Surveys*, June 1999, pp. 117-157.
4. E. Kovacs, K. Rohrle, and M. Reich, "Integrating Mobile Agents into the Mobile Middleware," *Proc. Mobile Agents Int'l Workshop*, Springer-Verlag, Berlin, 1998, pp. 124-135.
5. D. Kotz et al., "Agent TCL: Targeting the Needs of Mobile Computers," *Internet Computing*, July/Aug. 1997, pp. 58-67.

6. S. Lipperts and A. Park, "An Agent-Based Middleware: A Solution for Terminal and User Mobility," *Computer Networks*, Sept. 1999, pp. 2053-2062.
7. D. Lange and M. Oshima, *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley Professional, Boston, Mass., 1998.
8. A. Tripathi et al., "Distributed Collaborations Using Network Mobile Agents," *Symp. Agent Systems and Applications/Mobile Agents*, Springer-Verlag, Berlin, 2000, pp. 126-138.
9. P. Bellavista, A. Corradi, and C. Stefanelli, "Protection and Interoperability for Mobile Agents: A Secure and Open Programming Environment," *IEICE Trans. Comm.*, May 2000, pp. 961-972.
10. D. Milojicic et al., "MASIF: the OMG Mobile Agent System Interoperability Facility," *Proc. Mobile Agents Int'l Workshop*, Springer-Verlag, Berlin, 1998, pp. 50-67.
11. S. Lazar, I. Weerakoon, and D. Sidhu, "A Scalable Location Tracking and Message Delivery Scheme for Mobile Agents," *IEEE Int'l Workshop on Enabling Technologies*, IEEE Press, Piscataway, N.J., 1998, pp. 243-248.

***Paolo Bellavista*** *is a PhD student in computer science engineering at the University of Bologna. His research interests include distributed computing, distributed objects, mobile agents, network and systems management, adaptive multimedia systems, and mobile computing. He received a Laurea in electronic engineering from the University of Bologna. He is a student member of the IEEE, the ACM, and the Italian Association for Computing. Contact him at pbellavista@deis.unibo.it.*

***Antonio Corradi*** *is a full professor of computer science at the University of Bologna. His research interests include distributed systems, object and agent systems, network management, and distributed and parallel architectures. He received an MS in electrical engineering from Cornell University. He is a member of the IEEE, the ACM, and the Italian Association for Computing. Contact him at acorradi@deis.unibo.it.*

***Cesare Stefanelli*** *is an associate professor of computer science at the University of Ferrara. His research interests include distributed and mobile computing, mobile code, network and systems management, and network security. He received a PhD in computer science from the University of Bologna. He is a member of the IEEE and the Italian Association for Computing. Contact him at cstefanelli@ing.unife.it.*