The Ubiquitous Provisioning of Internet Services to Portable Devices

Advances in mobile telecommunications and device miniaturization call for providing both standard and novel location- and context-dependent Internet services to mobile clients. Mobile agents are dynamic, asynchronous, and autonomous, making the MA programming paradigm suitable for developing novel middleware for mobility-enabled services.

Paolo Bellavista and Antonio Corradi
Università di Bologna

Cesare Stefanelli
Università di Ferrara

The diffusion of wireless connections in home and office computing environments along with the proliferation of portable devices present new scenarios for service provisioning.[1] We must not only extend access to traditional Internet services to mobile users and devices but also develop new classes of location-dependent services. In addition, such provisioning to portable devices must consider the devices' strict limitations on hardware and software characteristics and their wide heterogeneity. Consider a tourist using a portable device to learn about a nearby piece of art. Already available Web services can provide detailed multimedia tourist information, which then must be downscaled to fit the device's bandwidth and visualization capabilities. In addition, tourists will want automatic filtering of Web information, depending on their position and specific interests.

These requirements demand novel support functions capable of transparently and dynamically adapting services to client location, user preferences, and device characteristics. Portable devices require middleware that exhibits nontraditional properties, such as location awareness and context adaptation, and that supports their accessibility to traditional and new services. Moreover, portable devices should be able to dynamically load and discard the client-side middleware and service components only when needed.

Our middleware dynamically extends the Internet infrastructure to accommodate portable devices. The main idea is to dynamically deploy middleware components that act over the fixed network on behalf of users and devices. We have designed and implemented components as mobile agents (MAs) to achieve mobility, asynchronicity, and autonomy.

Middleware guidelines for service provisioning to portable devices

Service provisioning to portable devices raises several technical challenges.

On the one hand, we must address issues related to portable device mobility. Recent research efforts have produced network-layer solutions to provide terminal connectivity in mobile scenarios.[2] However, relevant mobile computing issues exist—such as rapid service deployment, configuration, tailoring, security, and interoperability—that call for middleware solutions at higher levels of abstraction, including at the application level.[3] In particular, mobility-enabled naming solutions are crucial. Portable devices usually move among localities in an unpredictable way, without any static knowledge about the locally available resources and services. They should then be able to transparently connect to previously unknown resources and services. Mobile computing middleware should support the dynamic discovery of resources and services by imposing limited knowledge on the client side. It

1536-1268/02/$17.00 © 2002 IEEE

Discovery Solutions

Researchers in industry and academia have recently investigated how mobile users and terminals should dynamically retrieve and interact with the resources and service components that are available in a network locality, without assuming a deep knowledge at the client side. We usually identify these solutions as *discovery services*. Different discovery suites are available, each offering different technical approaches, features, and flavors.

Among the commercial solutions, the Bluetooth Service Discovery Protocol and the UPnP Simple Service Discovery Protocol represent simple, effective solutions at a low level of abstraction. The IETF Service Location Protocol and the Salutation suite are examples of more complex and articulated architectures of middleware components, with rich functions to query advertised services by attributes. The Jini solution is particularly interesting because, by assuming a language homogeneity, it lets us not only advertise and discover service components but also distribute information about how to access and use them (via interface proxy objects that dynamically migrate toward the clients).[1] However, Jini requires that discovery clients either run a Java Virtual Machine or associate with fixed docking stations, statically predetermined and preinstalled on the fixed network infrastructure, which runs a JVM on their behalf.

The proliferation of discovery research activities confirms that current commercial solutions do not sufficiently address all portable devices' needs. The DEAPspace system investigates completely decentralized discovery solutions, specifically suited to wireless ad hoc networks.[2] The Dynamic Mobility Agent proposal organizes network localities in a hierarchical way to achieve global scalability.[2] The Jini Surrogate project fills the gap between Jini discovery and roaming devices with no JVM. It specifies Jini-enabled surrogate components acting as Jini docking stations, which even non-Java portable devices can dynamically retrieve using low-level Bluetooth discovery (see http://developer.jini.org).

Discovery solutions are a basic building block of a mobile computing middleware, but they are not enough to provide location- or context-dependent services for portable devices. We need novel programming paradigms and technologies that can overcome the limits of the traditional client-server model when applied to mobile computing scenarios. In particular, several recent experiences suggest mobile code technologies—in particular, mobile agents—in the design and implementation of mobile computing middleware, because most MA requirements coincide with mobility ones.[3,4]

REFERENCES

1. G.G. Richard III, "Service Advertisement and Discovery: Enabling Universal Device Cooperation," *IEEE Internet Computing*, vol. 4, no. 5, Sept./Oct. 2000, pp. 18–26.
2. S.K.S. Gupta et al., "An Overview of Pervasive Computing," *IEEE Personal Comm.*, vol. 8, no. 4, Aug. 2001, pp. 16–59.
3. E. Kovacs, K. Rohrle, and M. Reich, "Integrating Mobile Agents into the Mobile Middleware," *Mobile Agents Int'l Workshop* (MA 98), Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1998, pp. 124–135.
4. S. Lipperts and A. Park, "An Agent-Based Middleware: A Solution for Terminal and User Mobility," *Computer Networks*, vol. 31, no. 19, Aug. 1999, pp. 2053–2062.

should also support easy use of and access to highly heterogeneous resources.

On the other hand, requirements stemming from specific hardware and software characteristics of portable devices exist—characteristics such as strictly limited resources and high heterogeneity. Portable devices have limited processing power, memory, and file system capabilities, and they are unsuitable for accessing traditional Internet services designed for fixed networks. The middleware should thus dynamically tailor (usually by downscaling) services to the client access device's specific characteristics. In addition, when portable devices need application-specific clients, their memory limitations require dynamically deploying client components (which we can install on portable devices) only when needed and then automatically discarding them after service provisioning. Moreover, current portable devices exhibit a high heterogeneity of hardware and software capabilities, operating systems, and supported network technologies. This heterogeneity not only makes it harder to provision different and statically tailored services, it also significantly increases configuration hassles for device users.

Middleware solutions should automate device configuration management as much as possible to free users from knowing and interacting with the implementation details specific to different devices. This scenario calls for novel middleware solutions that, unlike traditional middleware for Internet services, are aware of both location and context information and propagate this visibility up to the application level.

We define *location* as the property relating to the physical position of users, devices, resources, and service components. Location transparency can help create the high-level design of a fixed network's traditional services. However, it clashes with the development and deployment of services in mobile computing scenarios, where low-level components, such as the naming system, should propagate the location's explicit visibility up to the application level. In fact, mobile computing requires performing management operations on service provisioning, such as rebinding to locally available resources. These operations are typically at the appli-

cation level and relative to the current client location. Location visibility in a mobile computing environment requires the support of enhanced naming solutions that can effectively keep track of mobile entities. In addition, naming systems should integrate different support services with different scopes and efficiency, such as directory and discovery, to better suit the different requirements of applications that have local or global visibility.[3]

Portable devices should also be able to bind components to a set of resources and services that represent the current context. The notion of *context* involves *logical* properties—such as personal preferences, current session state, and history of past interactions—and *physical* properties—such as device profiles of characteristics and possible colocation of resources and access devices. A notable example is the set of resources and services that a user can access as requested by the preference profile, independent of the current point of attachment (such as the Virtual Home Environment[4]). Portable devices usually move among different network localities during service provisioning. Consequently, a mobile computing middleware should facilitate the devices' dynamic binding to new contexts. Note that a context can also dynamically change in response to modifications in the distributed system, such as when a service component fails or when a new resource becomes available.

Mobile agents for service provisioning to portable devices

To effectively design and deploy middleware with location and context visibility, we must consider technologies suitable for mobility. The "Discovery Solutions" sidebar presents state-of-the art discovery solutions for mobility-enabled naming. Here, we show how MAs can support services in mobile computing scenarios.

MA technology can significantly help us realize a distributed and decentralized infrastructure of proxies that work on behalf of the devices and that are hosted by the fixed network. MA-based proxies can follow a device's movements during service provisioning by maintaining the session state. Proxies can also automatically install their code in any newly visited network locality (but only when needed). In addition, MA adoption achieves the crucial properties of dynamicity, asynchronicity, autonomy, and location or context awareness. The telecommunications domain has also recognized MA's applicability in mobile computing. The Telecommunications Information Networking Architecture consortium has promoted the specification of novel middleware for personal mobility, integrating TINA access sessions with MAs.[5]

Mobile computing middleware offers dynamicity by letting new components and protocols modify and extend the fixed network infrastructure. This supports device accessibility and helps adapt services to evolving client requirements, even during service provisioning. Dynamic code distribution, typical of MAs, is crucial when dealing with portable devices because of their limited hardware and software characteristics and their heterogeneity. In fact, service providers cannot statically predict the versions of services suitable for all the access devices their customers use. Furthermore, they must be able to install or discard service components based on need. MAs can effectively play the role of data processors, migrating when necessary to install service components and tailor services by performing filtering and transcoding operations.[6]

Mobile computing can also take advantage of asynchronicity and autonomy between user requests (or device operations) and their execution. For instance, wireless connections impose strict constraints on available bandwidth and communication reliability and force portable devices to minimize their connection time. The MA paradigm does not assume continuous network connectivity; rather, it expects connections to last only for the time needed to inject agents from mobile clients to the fixed network. Agents are autonomous and can carry on services even after the launching users or devices disconnect. They can then return service results when the user or device reconnects.[7,8]

In addition, mobile computing middleware should provide application developers with location and context visibility so they can design location- and context-aware services. Location and context awareness are typical of the MA programming paradigm, where this visibility drives MA mobility decisions and is propagated up to the application level to enable design, deployment, and management choices depending on dynamic conditions.

MA platforms have investigated and proposed solutions to achieve other relevant middleware properties, such as security and interoperability. Even if not specific to the mobile computing domain, the availability of these solutions is important and can significantly leverage the diffusion of MA-based middleware to support Internet services.[3]

Due to the MA technology's novelty, few MA environments have been used to implement mobile computing middleware. The ACTS OnTheMove project[7] has adapted an existing MA system with a Mobile Application Support Environment to provide a gateway for mobility between fixed and wireless networks. Dartmouth Agent TCL can implement agents in different languages (TCL, Java, and Scheme) and provide a docking station in charge of forwarding agents and messages to mobile devices.[9] The Discovery MA system implements an infrastructure that notifies all interested agents of distributed events, such as the mobile device's connection and disconnection to support terminal mobility.[10] Other MA proposals concentrate on user profiling and profile-based Virtual Home Environments, by exploiting naming solutions with global visibility such as directory services.[8] To our knowledge, apart from our approach, Grasshopper is the only MA platform addressing the issues of enabling portable devices to access tailored Internet services. Grasshopper provides a specific Micro Edition version for portable devices with either Personal Java or Java 2 Micro Edition as the core technology at the basis of the Enago platform suite for mobile computing (see www.ikv.de/e0100en_platform.php).

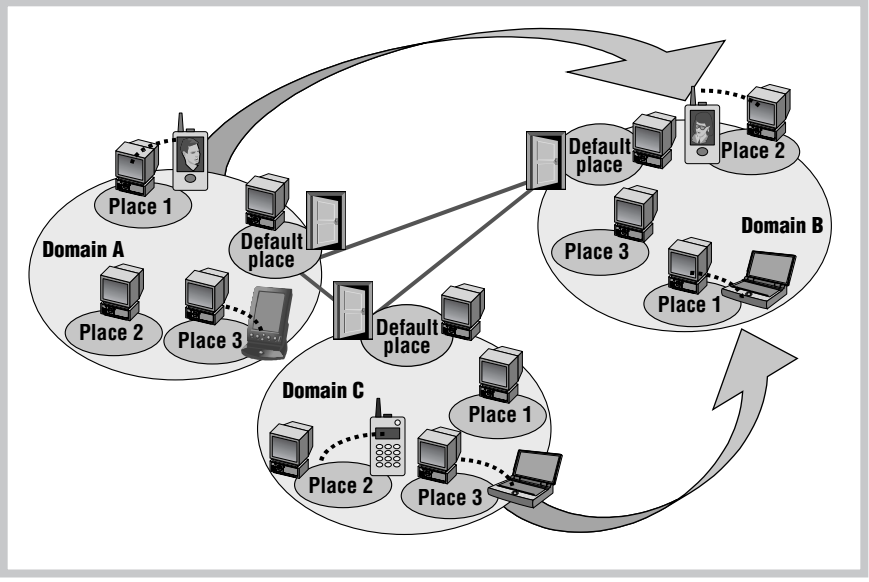

Figure 1. Mobile devices moving among secure and open mobile agent (SOMA) network localities.

The agent middleware for mobile devices

We have designed and implemented a middleware solution that enables service provisioning to portable devices with limited hardware and software characteristics and wired or wireless connectivity. We built it on top of the secure and open mobile agent (SOMA), a Java-based MA platform with a rich set of facilities for communication, migration, naming, persistency, security, and interoperability (http://lia.deis.unibo.it/Research/SOMA).[3,11] It dynamically extends the fixed network infrastructure to provide portable devices with both traditional and new location-dependent Web services. In addition, it exploits SOMA-based support for user and terminal mobility to accommodate portable devices that cannot host a full Java Virtual Machine.

Our middleware also tailors service provisioning to the characteristics of the access devices, as specified in the corresponding profiles, and performs service configuration and management operations that support a wide variety of highly heterogeneous client devices. In particular, it provides any portable device with an MA-based companion, called the *shadow proxy*, and with application-specific MA-based processors, called *service adapters*.

The Jini discovery solution (see the "Discovery Services" sidebar) also recognizes the centrality of middleware proxy migration toward clients to simplify and enhance service access. However, unlike Jini proxies, SOMA proxies and adapters can move by carrying both their code and their reached execution state. This lets the SOMA middleware components maintain and migrate session information when following portable device movements during service provisioning.

Architecture and implementation insights

The definition of flexible network localities is crucial for developing and deploying Internet services to portable devices. SOMA offers locality abstractions to describe any kind of interconnected system, from simple Intranet LANs to the Internet (see Figure 1). Any node hosts at least one place for agent execution; domain abstractions group several places and correspond to either fixed or wireless network localities. In each domain, a default place is in charge of interdomain routing. SOMA locality abstractions let us define loosely coupled localities organized in a hierarchical way. We have provided each SOMA domain with a local discovery service with intradomain visibility scope to enhance the SOMA naming scalability in large-scale deployment scenarios.[3]

The SOMA-based middleware for portable devices consists of the components in Figure 2: shadow proxies, service adapters, device-specific clients, the Portable Device Lookup Service (PDLS), and the Profile Manager Service (PMS).

Shadow proxies. Shadow proxies are application-independent middleware components that represent portable devices on the fixed network. They smooth problems due to intermittent device connections and resource limits and retrieve the profile of their companion devices' characteristics and of their current users' preferences. The profile information drives the service discovery at the PDLS. In addition, shadow proxies follow their portable devices in their movements among different SOMA domains. They carry the reached service state and make it possible to migrate service sessions dynamically. The shadow proxy's migration is triggered by its companion device's reconnection to a new SOMA domain. If the device does not reconnect for an interval greater than a specific threshold, the middleware automatically garbage-collects the associated proxy. The same occurs if the proxy can't reach the new connection domains due to network partitioning.

We currently associate one shadow proxy with each portable device, with a 1-to-1 mapping. It is also possible to implement group shadow proxies in charge of managing a set of portable devices with synchronization constraints on mobility and service provisioning. Our middleware implements the shadow proxy as a mobile agent running on a place in the SOMA domain where the device is currently located. Several shadow proxies for different devices can execute concurrently on the same node without interference because the SOMA platform provides isolated execution environments with separate security domains for the different agents.

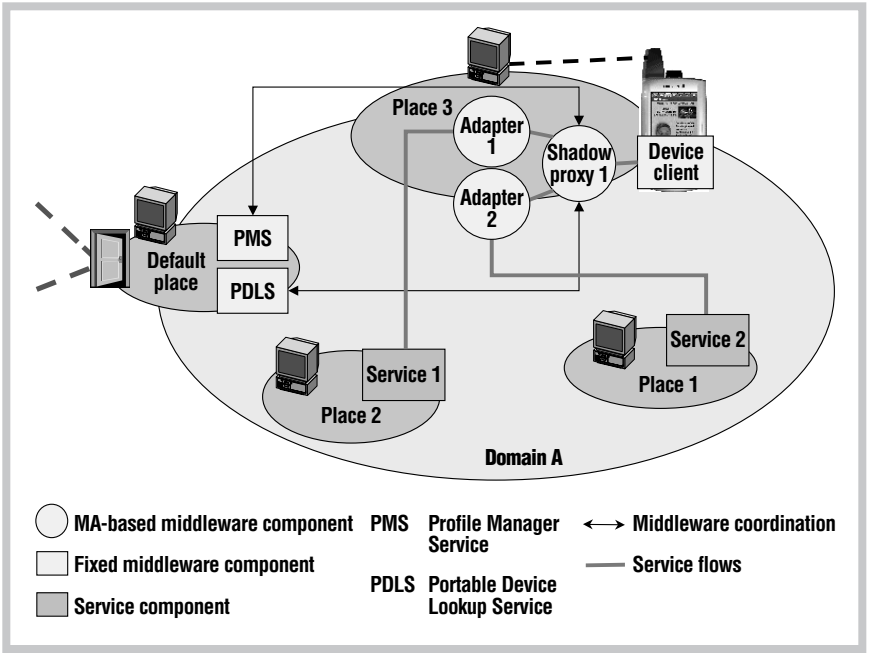

Figure 2. Portable device integration in the SOMA-based mobile computing middleware.

Service adapters. Service adapters are application-specific middleware components in charge of performing data transformations, depending on user or device profiles. Several different adapters can concurrently operate for one single shadow proxy to carry on parallel service requests from the same portable device. Our middleware implements service adapters as SOMA agents that follow their shadow proxy's movements. The default choice is to automatically migrate all adapters jointly with the proxy for which they work. We can also specify different mobility policies for the adapters in response to the associated proxy's migration, such as "immediately terminate the adapter" (the proxy then rebinds to new adapters in the new destination) or "maintain adapter persistence in the locality until the service session ends" (this saves processed service results on local persistent storage). In the latter case, for example, an adapter could filter location-dependent information that the proxy is interested in by asynchronously collecting the information and could deliver it back when the device reconnects in that locality.

We have implemented two different families of adapters: *filters* and *transcoders*. Filters can recognize and discard parts of service results whenever the client device cannot support their visualization (see the case study in the following section). Transcoders can operate even complex transformations on service results, such as HTML-to-WML conversion and multimedia format transcoding.[6]

Device-specific clients. Device-specific clients are the only middleware components that run in portable devices. These clients announce when the device enters or exits a network locality, ask shadow proxies for services, and receive the adapted service results. We have implemented three different types of lightweight device-specific clients. We based the first client on the J2ME/CLDC/MIDP suite for Palm devices with either USB or modem connectivity. The second one runs on top of the Java2 Standard Edition over Compaq personal digital assistants with Wireless LAN support. The third exploits the wireless Bluetooth discovery and is written in C within the Ericsson Bluetooth Application and Training programming environment.

Portable Device Lookup Service. PDLS and PMS are per-domain infrastructure components. PDLS is responsible for sensing when a portable device enters its SOMA domain and managing tailored lookup requests. Triggered by device arrival, PDLS asks the SOMA mobility-enabled naming whether the shadow proxy for that device is already running somewhere in the global system. SOMA naming locates mobile agents and devices based on *care-of* mechanisms. Any traceable mobile agent has its care-of (*agent home*) at the place of its instantiation. Similarly, any portable device has its care-of (*device home*) at the default place of its first profile registration domain; the device home keeps the information about the associated shadow proxy's agent home, if available. SOMA transparently updates the agents' homes at their migration and the devices' homes at their connection. Mobile agents and devices have GUIDs (globally unique identities) independent of their current position: GUIDs consist of the identifier of the corresponding home associated with a number unique in the home locality. For instance, a portable device has a GUID of the form (*DomainID, progNumber*), where *DomainID* is its device home's address. (Other details about the SOMA mobility-enabled naming appear elsewhere.[3])

After interrogating the SOMA naming, if there is an already active shadow proxy, PDLS triggers the proxy migration to its network locality. Otherwise, PDLS instantiates a new local shadow proxy for the device. When shadow proxies ask for services, PDLS does not provide a reference to the service to carry out the request (the usual lookup service behavior); rather, it provides a reference to an adapter that acts as the intermediate between the shadow proxy and the actual service component. PDLS is built on top of the Jini Reggie reference implementation of the lookup server and significantly extends its functions to suit the specific needs of service binding to portable devices. In particular, it additionally considers the user and device profiles that the shadow proxy provides, identifies the needed service adapter,

binds it to the requested service component, and finally triggers the adapter migration to the requesting shadow proxy's location. If no adapters are compatible with the specified profiles—for instance, if the service request results in a multimedia flow while only text is supported on the requesting device—PDLS sends a service unavailability message to the device client. We specify service components and adapters according to the XML-based Web Services Description Language.[12]

Any SOMA domain models a different area and maintains information about the local movie theatres. When a portable device enters a domain, SOMA either instantiates its shadow proxy or moves it to the place where the device is currently attached. In the case of new instantiation, the middleware transparently downloads the proxy code from the repository at the local default place and activates it; the proxy authenticates the user connected at the device, recognizes the type of portable device, and asks PMS for user and device profiles. In the case of proxy migration, the middleware moves both the code and state of the proxy from the previous point of attachment. Note that proxies do not need to transfer their code at any migration because SOMA places can cache incoming Java classes and trigger code migration only when the code is not available in the locality. Then, when the device requests the WOIT service, the proxy interrogates PDLS by providing the retrieved user and device profile information. PDLS answers by triggering the migration of a suitable WOIT service adapter to the proxy; similar to the proxy code, the WOIT adapter code also migrates if a cached copy of it is not already present.

SOMA-based middleware lets portable devices access an Internet service without modifying its design and implementation.

Profile Manager Service. PMS maintains profiles of supported devices and registered users. It implements a partitioned and partially replicated directory service specialized for profiles. It keeps local partial copies of profile information and coordinates with PMSs in other SOMA domains to provide shadow proxies with the visibility of any profile registered in the system. It also expresses user and device profiles according to the XML-based Composite Capability/Preference Profiles that the World Wide Web Consortium promotes. This permits a concise profile specification by identifying only the differences from standard default properties and by merging profile fragments that are dynamically retrieved even from different Web sites—for example, from profile repositories of device vendors.

The "What's On In Town?" service

To describe how the middleware components interwork to support portable devices, we present a simple location- and context-dependent service called "What's On In Town?" The WOIT service provides tailored and personalized information about movies showing at cinemas in the current locality of the portable access device.

We have implemented three simple WOIT service adapters, which perform filtering and ordering operations for the three different categories of supported portable devices:

- The adapter for the J2SE-based client simply receives the information from the server and passes it as it is to the proxy.
- The adapter for the J2ME-based client can recognize streaming flows, such as audio or video trailers, and discard them from service results.
- The adapter for the Bluetooth-enabled device client discards not only streams but also any fixed image to return only ASCII information to the proxy.

Any service adapter, independent of its filtering functions, orders the movie list according to profiled user preferences (such as thriller, drama, or comedy). Therefore, in the WOIT prototype, the device profile helps the proxy select the most suitable adapter, while the user profile affects the ordering criteria of service results.

SOMA-based middleware lets portable devices access an Internet service without modifying its design and implementation. With WOIT, the service component is a standard HTTP server that maintains textual information about the movies currently showing in the area and has fixed images of their posters and their multimedia trailers. In WOIT, we express Web pages in XML according to the standard Dublin Core Metadata specifications.

In addition, you do not need to keep the portable device connected to the network infrastructure while its shadow proxy obtains the suitable adapter and, from it, the tailored service results. In fact, the proxy immediately forwards the results to the device if connected; otherwise, it caches results locally while waiting for the device to reconnect, either in the same domain or outside that domain. In the latter case, the proxy migrates to the new domain of attachment and asks whether the device client is still interested in receiving results about the previous area or if it prefers to relaunch the service for the new locality.

Recent research on middleware solutions for mobile computing confirms the suitability of mobile code programming paradigms. Among them, MAs facilitate the design and implementation of extensible middleware, where behavior and session state can dynamically move where and when needed to follow user and device mobility.

Even if researchers start to recognize that MA technology is suitable for supporting user and terminal mobility, not many MA systems have been exploited in the implementation of mobile computing middleware. In particular, our solution with mobile middleware components hosted in the fixed network originally shows the MA-based approach's applicability to dealing with client devices with strict limitations on hardware and software characteristics. In fact, as far as we know, our middleware is the first MA-based solution that lets you integrate portable devices with no JVM. This aspect is crucial because, on the one hand, most current portable devices do not host any version of the JVM. On the other hand, even when devices support limited JVM versions—such as the J2ME's Sun K virtual machine—it is usually impossible to run MAs directly on them because of the severe rigidity of their class loading functions. For instance, J2ME does not instantiate programmer-defined class loaders, as Java-based MA platforms commonly do.

For the sake of simplicity, we only presented the WOIT example to show the SOMA-based middleware components at work. However, we are extensively using our middleware to develop more complex services in several application domains, including museum visitor assistance and video-on-demand distribution. The Virtual Museum Assistant retrieves information about the artwork exposed in a currently visited museum room and adapts its presentation depending on the current context, such as device characteristics, visitor expertise level, and preference profile. The ubiQoS prototype[6] provides video-on-demand services with differentiated levels of quality by tailoring, at service negotiation, the offered quality depending on access device capabilities. At provision time, it adapts the offered quality to modifications in the availability of involved distributed resources. P

ACKNOWLEDGMENTS

The Italian Ministero dell'Istruzione, dell'Università e della Ricerca, supported this work in the framework of the Project Musique: Infrastructure for QoS in Web Multimedia Services with Heterogeneous Access.

REFERENCES

1. G. G. Richard III, "Service Advertisement and Discovery: Enabling Universal Device Cooperation," *IEEE Internet Computing*, vol. 4, no. 5, Sep./Oct. 2000, pp. 18–26.
2. C. Perkins, "Autoconfiguration Plug & Play Internet," *IEEE Internet Computing*, vol. 3, no. 4, July/Aug. 1999, pp. 42–44.
3. P. Bellavista, A. Corradi, and C. Stefanelli, "Mobile Agent Middleware for Mobile Computing," *Computer*, vol. 34, no. 3, Mar. 2001, pp. 73–81.
4. L. Bos and S. Leroy, "Toward an All-IP-Based UMTS System Architecture," *IEEE Network*, vol. 15, no. 1, Jan./Feb. 2001, pp. 36–45.
5. H. Jormakka et al., "Agent-based TINA Access Session Supporting Retailer Selection in Personal Mobility Context," *Proc. Int'l Telecommunications Information Networking Architecture Conf.* (TINA 99), IEEE Press, Piscataway, N.J., 1999, pp. 68–76.
6. F. Baschieri, P. Bellavista, and A. Corradi, "Mobile Agents for QoS Tailoring, Control and Adaptation over the Internet: The UbiQoS Video on Demand Service," *Proc. 2nd IEEE Int'l Symp. Applications and the Internet* (SAINT 02), IEEE CS Press, Los Alamitos, Calif., 2002, pp. 109–118.
7. E. Kovacs, K. Rohrle, and M. Reich, "Integrating Mobile Agents into the Mobile Middleware," *Mobile Agents Int'l Workshop* (MA 98), Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1998, pp. 124–135.
8. S. Lipperts and A. Park, "An Agent-Based Middleware: A Solution for Terminal and User Mobility," *Computer Networks*, vol. 31, no. 19, Aug. 1999, pp. 2053–2062.
9. D. Kotz et al., "Agent TCL: Targeting the Needs of Mobile Computers," *IEEE Internet Computing*, vol. 1, no. 4, July/Aug. 1997, pp. 58–67.
10. S. Lazar, I. Weerakoon, and D. Sidhu, "A Scalable Location Tracking and Message Delivery Scheme for Mobile Agents," *IEEE Int'l Workshop Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE 98) IEEE CS Press, Los Alamitos, Calif., 1998, pp. 243–248.
11. P. Bellavista, A. Corradi, and C. Stefanelli, "Protection and Interoperability for Mobile Agents: A Secure and Open Programming Environment," *IEICE Trans. Comm.*, vol. E83-B, no. 5, May 2000, pp. 961–972.
12. F. Curbera et al., "Unraveling the Web Services: An Introduction to SOAP, WSDL, and UDDI," *IEEE Internet Computing*, vol. 6, no. 2, Mar./Apr. 2002, pp. 86–93.

the AUTHORS

Paolo Bellavista is a research associate of computer engineering at the University of Bologna. His research interests include mobile agents, mobile computing, network and systems management, location- and context-aware services, and adaptive multimedia systems. He received a PhD in computer science engineering from the University of Bologna. He is member of the IEEE, ACM, and Italian Association for Computing. Contact him at the Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, Viale Risorgimento, 2-40136 Bologna, Italy; pbellavista@deis.unibo.it.

Antonio Corradi is a full professor of computer engineering 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, ACM, and Italian Association for Computing. Contact him at the Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, Viale Risorgimento, 2-40136 Bologna, Italy; acorradi@deis.unibo.it.

Cesare Stefanelli is an associate professor of computer engineering 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 Italian Association for Computing. Contact him at the Dipartimento di Ingegneria, Università di Ferrara, Via Saragat, 1-44100 Ferrara, Italy; cstefanelli@ing.unife.it.