Context-Aware Middleware for Resource Management in the Wireless Internet

Paolo Bellavista, *Member*, *IEEE Computer Society*, Antonio Corradi, *Member*, *IEEE*, Rebecca Montanari, *Member*, *IEEE*, and Cesare Stefanelli, *Member*, *IEEE*

Abstract—The provisioning of Web services over the wireless Internet introduces novel challenging issues for service design and implementation: from user/terminal mobility during service execution, to wide heterogeneity of portable access devices and unpredictable modifications in accessible resources. In this scenario, there are frequent provision-time changes in the context, defined as the logical set of accessible resources depending on client location, access terminal capabilities, and system/service management policies. The development of context-dependent services requires novel middlewares with full context visibility. We propose a middleware for context-aware resource management, called CARMEN, capable of supporting the automatic reconfiguration of wireless Internet services in response to context changes without any intervention on the service logic. CARMEN determines the context on the basis of metadata, which include declarative management policies and profiles for user preferences, terminal capabilities, and resource characteristics. In addition, CARMEN exploits the mobile agent technology to implement mobile middleware components that follow the provision-time movement of clients to support locally their customized service access. The proposed middleware shows how metadata and mobile agents can favor component reusability and automatic service reconfiguration, by reducing the development/deployment complexity.

Index Terms—C.2.8.e mobile computing/support services, J.8.l Internet applications/middleware, J.9.a mobile applications/location-dependent and sensitive, C.2.8.d mobile computing/mobile environments.

1 Introduction

Let us start by considering a service scenario which will be more and more usual in the next days. Alice with her IEEE 802.11b palmtop is willing to access a Mobile News Service (MNS) that permits the reading and browsing of newspaper data resources available in the fixed Internet. For instance, MNS allows Alice's palmtop to download the news to read during the flight from an info-station at the boarding gate; after the plane lands, the palmtop can automatically reconnect to a new info station at the arrival gate to obtain updated news, as well as locality-dependent information, e.g., weather and traffic reports. The scenario can require supporting not only the wireless connection of Alice's palmtop to the needed resources in the fixed network via different IEEE 802.11b access points [1], but also the automatic requalification of accessed resources depending on the client location, the currently enforced management policies in the hosting locality, and Alice's personal preferences. For instance, Alice is also interested in accessing locality-dependent information but only if in English; in the case of only Japanese news locally available, she would like to access either remote data resources or a local Japanese-to-English service component automatically translating the local news.

Then, let us follow Alice who leaves her palmtop at home and goes to a gym in the late afternoon. Alice loves doing her yoga exercises while listening to music from her Bluetooth-enabled micro mp3 reader. The reader has a little text-only display to visualize playing song information and also short flowing messages. Also, in the gym, Alice would like to access MNS via Bluetooth connectivity, but this time only to receive short news excerpts tailored to her interests and subscribed topics, e.g., to be notified of sport results and abrupt modifications in the quotation of her stock exchanges.

The MNS scenario exemplifies how it is crucial to consider both mobility and heterogeneity in service provisioning to wireless portable devices. On the one hand, mobility requires tracking clients, suggests connecting them to the most suitable resources, either local or remote, and motivates the provisioning of new location-dependent services [2]. In addition, mobility stimulates the possibility to continue service operations also asynchronously with regards to clients, for instance, when performing a time-consuming complex query on geographically distributed news resources, and to maintain the session state between client disconnection and successive reconnection, possibly in a different access locality, for instance, in an MNS version taking into account the downloaded news to propose only unread new ones. On the other hand, the high heterogeneity of portable access devices forces to consider not only the support of different forms of wireless connectivity, but also the change in the logical set of accessible resources during service provisioning and the corresponding need for

- *P. Bellavista, A. Corradi, and R. Montanari are with the Department of Electronics, Computer Science, and Systems (DEIS), University of Bologna, 40136 Bologna, Italy. E-mail: {pbellavista, acorradi, rmontanari}@deis.unibo.it.*
- *C. Stefanelli is with the Department of Engineering, University of Ferrara, 44100 Ferrara, Italy. E-mail: cstefanelli@ing.unife.it.*

Manuscript received 31 Dec. 2002; revised 6 July 2003; accepted 5 Aug. 2003. Recommended for acceptance by M. Oivo and M. Morisio.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 118785.

modifying service management decisions, e.g., dynamically tailoring the MNS results to a very limited access device by filtering only user-specific textual information.

MNS works in a common networking scenario where wireless solutions can extend the accessibility of the fixed Internet infrastructure via access points working as bridges between fixed hosts and wireless devices [3]. An exemplar case is the utilization of IEEE 802.11b access points to support the connectivity of Wi-Fi equipped laptops to a wired local area network [1]. We will indicate these integrated networks with fixed Internet hosts, wireless terminals, and wireless access points in between, as the *wireless Internet*.

The MNS scenario significantly changes several aspects of service provisioning because there is the need for full visibility of the service provisioning *context*, defined as the logical set of resources accessible to a client during a service session depending on several factors, such as client location, access device capabilities, management policies of the access locality, subscribed services, user preferences, and level of trust.

The flexible and effective management of context information is a complex and challenging issue per se [4] and is further complicated by the frequent context changes typical of service provisioning in the wireless Internet. Mobility forces to handle properly changes of client location, modifications in locally accessible resources, temporary disconnection, and changing network topology; users can change portable access devices, with different wireless technologies, even at runtime, thus forcing to consider very dynamic aspects also due to the client heterogeneity.

The handling of context information and of its modifications at provision time significantly increases the complexity and the costs of designing, developing, and deploying services in the wireless Internet, thus slowing down their widespread acceptance and diffusion. There is the need for nontraditional middleware infrastructures to support context-dependent services to wireless portable devices: Novel middleware solutions should achieve full context visibility, automate service reconfiguration depending on dynamic context changes, and provide high-level ways to specify context-aware behavior separately from application logic.

The paper proposes a novel middleware for context-aware resource management, called CARMEN (**C**ontext-**A**ware **R**esource **M**anagement **EN**vironment), capable of supporting the automatic reconfiguration of Web services for the wireless Internet in response to context changes, without any intervention on the service application logic. CARMEN allows service providers, system administrators, and final users to specify service management requirements at a high level of abstraction in terms of different kinds of metadata: declarative management policies for migration, binding and access control, and profiles for the description of user preferences, device capabilities, and service component characteristics. In addition, a distinguishing key feature of CARMEN is the exploitation of mobile middleware proxies that follow the provision-time movement of users, where and when needed, not only to support locally their service accessibility, but also to customize service provisioning, to maintain service session state, and to operate asynchronously with regards to temporarily disconnected clients. CARMEN implements mobile proxies in terms of Mobile Agents (MAs) because the MA programming paradigm is particularly suitable to achieve crucial middleware properties such as mobility, autonomy, asynchronicity, and location awareness [5].

The paper also presents the development and deployment of the CARMEN-based MNS. MNS clients can run over heterogeneous portable devices, from fully equipped laptops to very limited palmtops, with either Wi-Fi or Bluetooth connectivity. The CARMEN-based MNS shows the usability and the effectiveness of the proposed middleware and points out how the design and implementation of wireless Internet services is simplified and accelerated by the adoption of middleware solutions based on metadata and MAs.

The rest of the paper is organized as follows: Section 2 points out the crucial role of context-aware resource management in the wireless Internet and the suitability of adopting metadata for its support. Section 3 describes mobility and binding management issues, by proposing solution guidelines based on mobile middleware components. Section 4 and Section 5, respectively, present the different kinds of metadata exploited in the CARMEN middleware and its layered architecture, while Section 6 is devoted to describe the design and implementation of the CARMEN mobile proxies. Section 7 shows the implementation of MNS on top of CARMEN and reports some experimental results. Related work, conclusive remarks and directions of current work end the paper.

2 Context-Aware Service Provisioning

The wireless Internet scenario has several specific characteristics to be considered in service provisioning. Mobility of users and access devices is pushed to the extreme. Users can connect to the network from ubiquitous points of attachment and wireless portable devices can roam by maintaining continuous connectivity [6]. Frequent disconnections of users/devices are rather common operating modes that can occur either voluntarily to reduce connection costs and to save battery power or accidentally due to the loss of wireless connectivity. In addition, the wireless Internet exhibits a high degree of heterogeneity of both access devices (in terms of screen size/resolution, computing power, memory, operating system, and supported software) and networking technologies (IEEE 802.11b, the emerging IEEE 802.11g, Bluetooth, GSM, GPRS, and UMTS).

The distinctive features of the wireless Internet pose new challenges in retrieving and operating on distributed resources and undermine several assumptions of traditional service provisioning. The main impact is on the notion of context. Traditional service provisioning relies on a relatively static characterization of the context, where resource availability is independent of both the user current location and the access device properties (location and heterogeneity transparency); changes in the set of accessible resources are relatively small, rare, or predictable [7]. The location and

heterogeneity transparency leads to traditional monolithic solutions that tend to suffer from insufficient flexibility and dynamicity when applied to the wireless Internet.

Service provisioning in the wireless Internet requires the visibility not only of location information but also of other system-level data, such as access device characteristics. This information should be propagated up to the service level to dynamically determine the client context and to perform service configuration and delivery accordingly. In addition, mobility determines changes in the physical user location and in the consequently perceived context. Context variations can be very frequent, especially when using wireless portable devices. In the MNS example, Alice could connect with her palmtop at different info stations along the route from home to the airport to access updated information about traffic congestion in the proximity of her current location. Let us observe that context and location, though intimately related, are different notions: Two mobile devices may be at the same location but perceive different contexts because of their different capabilities, of their belonging to different administrative domains, or of their utilization by users with different preferences [7].

2.1 Metadata-Based Middleware Solutions for Context Management

The above considerations call for the design of novel middleware solutions to support context-aware service provisioning. Novel middleware should interact with the underlying execution environment to collect relevant information for context determination, e.g., current location of users/devices, state of resources, user preferences, and device characteristics. This information should be processed at provision time to identify the context and its evolution, and to propagate it up to the service level.

Let us note that the context changes due to the high dynamicity of the execution environment require proper management actions to adapt service provisioning. The wireless Internet mobility and heterogeneity make service management a very complex task, requiring novel appropriate methodologies and tools to flexibly specify which management actions to perform and in which runtime conditions and to promptly carry out the desired service reconfiguration. Reconfiguration requirements should be expressed at a high level of abstraction by cleanly separating service management from service logic. This separation of concerns is crucial to reduce the complexity of developing services for the wireless Internet and to favor rapid service prototyping, runtime configuration, and maintenance. In the MNS scenario, when Alice switches from the palmtop to the micro mp3 reader, the context change should determine the corresponding MNS management operations to dynamically tailor the news format to the new access device.

To support context awareness and to perform service management accordingly, we propose the adoption of metadata for representing both the context characteristics and the choices in service behavior at a high-level of abstraction, with a clean separation between service management and service logic.

Metadata can describe both the structure/meaning of the resources composing a system and the specification of management operations expressed at a high level of abstraction [8]. Among the different possible types of metadata, profiles and policies are considered of increasing interest and start to be widely exploited in open and dynamic distributed systems. Profiles represent characteristics, capabilities, and requirements of users, devices, and service components. Several research efforts are attempting to identify well accepted formats for the most common access devices and spreading standard profile adoption for expressing user needs/requirements; profile standardization is crucial for resource reusing and sharing in the open wireless Internet. Policies express the choices ruling system behavior, in terms of the actions subjects can/must operate upon resources. Policies are maintained completely separated from system implementation details and are expressed at a high level of abstraction to simplify their specification by system administrators, service managers, and even final users. Policy-based systems distinguish two different kinds of policies [9]: Access control policies specify the actions subjects are allowed to perform on resources depending on various types of conditions, e.g., subject identity and resource state; obligation policies define the actions subjects must perform on resources when specified conditions occur.

The effectiveness of the metadata adoption depends on the characteristics of the language used for metadata specification and of the runtime environment for the metadata support. Metadata specification should exploit declarative languages to accommodate users of different expertise, to simplify metadata reuse and modification, and to facilitate the analysis of potential conflicts and inconsistencies. The metadata runtime support should be responsible for metadata distribution/update and for policy activation/deactivation/enforcement, independently of service logic.

3 Resource Management for Mobility

Context changes force to consider the management issues of updating dynamically the binding of clients to resources. In the wireless Internet, user/device mobility is a very usual source of context change. For instance, a mobile client could bring the needed resources (or their copies) with itself, or it could discard the bindings to old resources upon migration and rebind to new suitable resources upon the arrival at the new destination. In general, it is possible to identify four different resource binding strategies when a mobile entity (ME) moves in the network:

- ***resource movement*** strategy. When one ME moves, its bounded resources are transferred along with it. This type of binding is possible only if the resource transfer is technically/semantically possible and requires handling properly the case of resource sharing between different clients.
- ***copy movement*** strategy. When one ME migrates, copies of its bounded resources are created and transferred along with it. This type of binding is permitted only if the resource copy is technically/semantically possible and may require to resolve

conflicts in case of concurrent modifications on multiple resource copies.

- ***remote reference*** strategy. This strategy does not move the resources and instead modifies the ME bindings after migration to refer remotely the resources. Any ME operation implies a network communication with the remote environments hosting the bounded resources.
- ***rebinding*** strategy. The ME movement triggers a rebinding to equivalent resources available in the new locality. This strategy typically applies to the case of by-type bindings [5] and is fundamental anytime an ME is interested in accessing resource instances that provide service contents depending on the instance location.

There is a wide variety of basic and heterogeneous implementation mechanisms that can help in realizing the above binding strategies, from solutions for resource retrieval, access and usage, e.g., the Bluetooth Service Discovery Protocol [10], the Service Location Protocol [11], and Jini [12], to mechanisms for the description of service components and the local/remote interaction with them, e.g., the Web Services Description Language (WSDL) [13], the Java Remote Method Invocation (RMI) [14], and the Service Object Access Protocol (SOAP) [15]. In addition to the choice of the most suitable binding mechanism, however, a very complex aspect is the design of solutions to dynamically change binding strategies depending on the management requirements of service providers, on the runtime conditions of the execution environment, on user preferences, and on access device properties. Traditional middleware approaches embed a static binding strategy within the service logic, thus limiting the flexibility of binding management [16]. Novel middlewares should support binding strategies defined and modified at provision time, depending on dynamic conditions. For instance, when Alice is at the airport waiting for her flight, she should use all the accessible MNS resources, either local or remote (remote reference binding). Before boarding the plane, she should copy the needed resources on her palmtop to work on them while disconnected (copy movement binding).

Several proposals recognize the importance of dynamically extending the wireless Internet infrastructure with proxies acting on behalf of (possibly disconnected) limited devices [17], [18], [19]. Proxies can perform several service management operations, e.g., disconnection support and service result caching. We consider it crucial that these proxies are mobile in order to follow the client movements and to offer the needed support only where and when needed. Adding mobility to proxies requires technologies to support dynamic code/state migration, mobility-enabled communication and coordination, and also flexible management of binding strategies. To this purpose, we claim the suitability of mobile code programming paradigms in general, and in particular of Mobile Agents (MAs) as the implementation technology of mobile proxies in the wireless Internet.

3.1 Mobile Agents for the Design of Mobile Proxies

The properties of mobility, autonomy, asynchronicity, and local resource exploitation typical of the MA programming paradigm are particularly suitable for the design and implementation of mobile proxies [20]. MAs can autonomously operate to carry on operations on needed resources even in case of temporary device disconnection and can migrate dynamically, either to follow device movements or to operate locally to the needed resources. With the adoption of MA-based proxies, wireless portable devices need limited network connectivity, for instance, only to inject in the fixed network the responsible proxies acting on their behalf.

In addition, the MA programming paradigm is typically location-aware, meaning that the location of the MAs is raised to the status of a first-class design concept. MAs can exploit the visibility of their execution environment to adapt their actions, primarily their migration, to the position of needed resources. In the case of mobile proxies, there is the necessity to modify proxy migration choices at runtime depending on user needs, network connectivity, etc. Mobility strategies should be specified at a high level of abstraction, with no need to modify the MA implementation.

Finally, MAs can provide full decentralization of management control, which is definitely important in global scenarios to achieve scalability and to avoid management bottlenecks. In particular, management decentralization is crucial when dealing with significant discontinuities in network resources, such as when passing from a wired attachment point to wireless connectivity, which can require performing system/service management operations locally to the discontinuity.

4 Metadata in CARMEN

CARMEN is a novel middleware for context-aware resource management that supports and facilitates the design, development, and deployment of context-dependent services for the wireless Internet. CARMEN allows service providers, system administrators, and final users to specify different kinds of metadata in a declarative way at a high level of abstraction. CARMEN metadata influence the dynamic determination of context and, consequently, the context-based service provisioning, without any intervention on the application logic, according to the design principle of separation of concerns.

CARMEN exploits two types of metadata: *profiles* to describe the characteristics of any resource modeled in the system, and *policies* to manage migration, binding and access control (see Fig. 1).

CARMEN profiles describe users, devices, service components, and sites. In particular, *user profiles* maintain information about personal preferences, interests, security requirements, and subscribed services for any CARMEN registered user. *Device profiles* report the hardware/software characteristics of the supported access terminals. *Service component profiles* describe the interface of available service components as well as their properties relevant for binding management decisions, e.g., whether a service

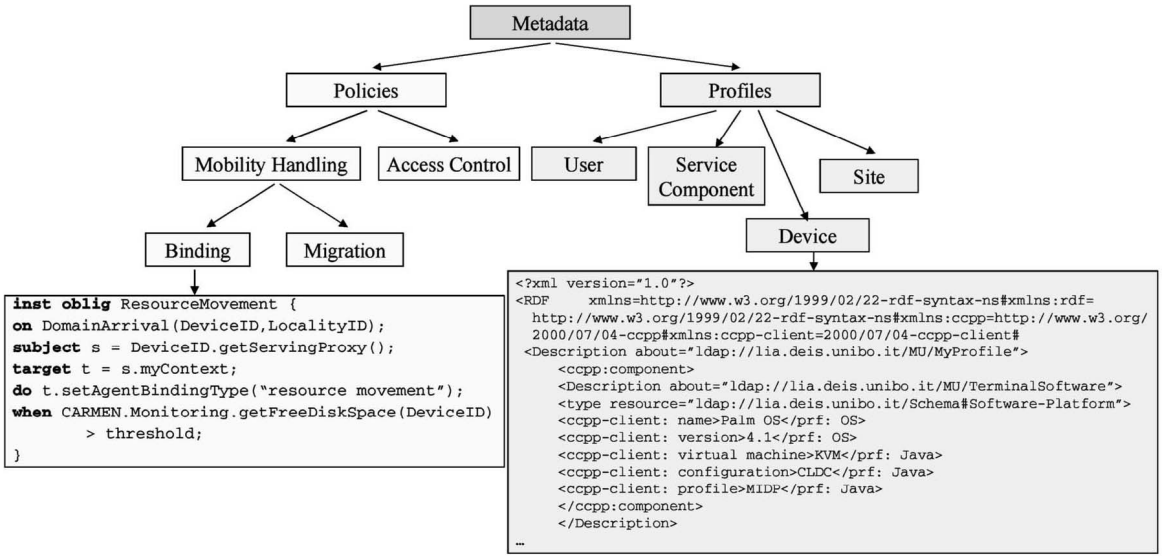

Fig. 1. CARMEN metadata: taxonomy and examples.

component can be copied and migrated over the network. *Site profiles* provide a resource group abstraction, by listing all the resources currently available at one CARMEN host.

CARMEN adopts XML-based standard formats for profile representation to deal with the Internet openness and heterogeneity: the World Wide Web Consortium Composite Capability/Preference Profiles (CC/PP) for user/device profiles [21], WSDL for the service component interface description [13], and the Resource Description Framework (RDF) for the site collections of resources [22]. For instance, Fig. 1 shows the CC/PP-compliant profile for a PalmOS device hosting the KVM/CLDC/MIDP software suite [23]. CARMEN profiles are stored in a partitioned and partially replicated directory service, compliant with the Lightweight Directory Access Protocol (LDAP) and specialized for profiles [20].

In addition to profiles, CARMEN expresses policies as high-level declarative directives. CARMEN distinguishes two types of policy metadata: *access control* policies to ensure secure resource usage and *mobility handling* policies to guide the middleware decisions in response to provision-time context variations.

Mobility handling policies, in their turn, include two different types of policies for dynamic context management, which derive from different management goals. On the one hand, CARMEN *migration policies* specify under which circumstances, where, and which middleware components and resources have to migrate triggered by the user movements. A specific type of migration policy are colocality ones describing the circumstances under which it is convenient to allocate a set of resources in the same site. In the MNS example, suppose that Alice is interested in reading the same political news from two opposite opinion tabloids to compare viewpoints. Colocating copies of the two tabloid resources in the current Alice's locality may be convenient to improve performance and to increase overall accessibility in case of network partitioning. On the other hand, CARMEN *binding policies* define when and which binding strategy to apply to update the set of needed resources after any change of client context. CARMEN supports all the four binding strategies introduced in Section 3 (resource movement, copy movement, remote reference, and rebinding). For instance, depending on the chosen binding policy, Alice's movement can either trigger the copy of a remote MNS resource to her new access locality or request the reconnection to an equivalent local resource, e.g., to access location-dependent MNS information.

CARMEN adopts the Ponder language for policy specification [24]. In particular, we use Ponder obligation policy types for the definition of mobility handling policies, and Ponder authorization policy types for access control. In the following, we focus only on obligation policies, central to the CARMEN context-aware resource management tasks, whereas readers can refer to [25] for details about Ponder authorization policies.

CARMEN obligation policies are expressed as declarative event-action-condition rules defining the actions that policy subjects must perform on target objects when specific events occur. Fig. 1 shows an example of a Ponder-based binding policy for selecting the resource movement strategy after the movement of a wireless portable device identified as `DeviceID`. In particular, the `ResourceMovement` policy states that, when `DeviceID` arrives at a new execution locality `LocalityID` (`on` clause), the `DeviceID` serving proxy (`subject` clause) should command its `myContext` object (`target` clause) to activate a resource movement binding strategy (`do` clause), if the device has enough free space on disk, as observed at runtime by the underlying CARMEN Monitoring facility (`when` clause).

5 The CARMEN Architecture

The CARMEN middleware is designed according to the layered architecture shown in Fig. 2. The Metadata Manager (MM) and the Context Manager (CM) compose the high-level middleware facilities. MM supports the specification, modification, checking for correctness, installation, and evaluation of the different kinds of profiles and policies

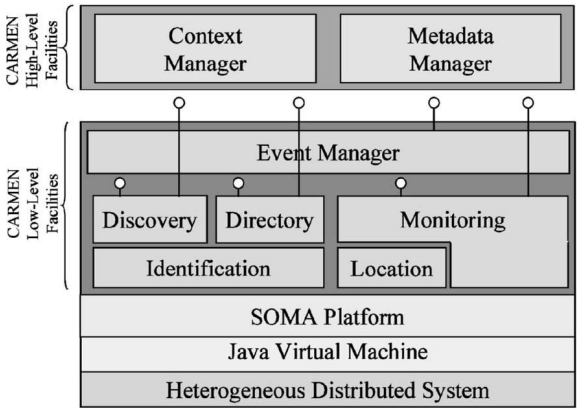

Fig. 2. The CARMEN layered architecture.

supported in CARMEN. CM dynamically determines the context of a CARMEN client, supports the accessibility of the resources included in the context, and manages resource bindings in case of context modifications. MM and CM are the crucial components of the proposed middleware and will be described more in detail in the following.

The CARMEN low-level facilities provides mechanisms and tools to address most common issues in context-aware service provisioning to wireless clients, such as a rich and articulated naming system (the *Identification*, *Discovery*, and *Directory* facilities) [20], a *Location* facility integrating heterogeneous tracking solutions for the different wireless technologies supported (IEEE 802.11b and Bluetooth), a *Monitoring* facility to observe indicators at both the application level and at the system one, in order to achieve full visibility of context changes [26], and the *Event Manager* (EM), presented in the following.

5.1 Metadata Manager

MM supports the specification of the different kinds of CARMEN metadata. It is in charge of supporting the specification/update of profile information, and of dynamically installing/enforcing policies for access control and mobility handling. In the following, we focus on the aspects related to mobility handling policies (see [16], [19] for further details about the management of the other metadata types). The enforcement of mobility handling policies involves the detection of changes in the operating environment (via the Monitoring and Location facilities), the notification of event occurrence to the interested policy subjects (via the EM facility), and the MM interpretation of enforced policies so as to activate the specified management actions.

MM is organized in three different logical modules: the Specification Module (SM), the Obligation Coordinator (OC), and the Obligation Enforcer (OE). SM exploits the tools developed within the Ponder project for editing, distributing, updating, removing, and browsing obligation policies [24]. In addition, it provides tools for transforming high-level policy specifications into platform-enforceable policy representations. In particular, SM generates individual Java policy objects for each Ponder obligation policy. When a new policy object is created, it is registered in the CARMEN Directory facility, stored in policy repositories, and distributed to the interested policy subjects, e.g., the CARMEN mobile proxies described in the following section.

The OC module coordinates the policy enforcement. It retrieves newly instantiated Java policy objects from the repositories and parses them to retrieve relevant information: events, subjects, targets, and actions. Then, on behalf of policy subjects, it registers the significant events to EM.

It is the OE module that actually enforces the policies. When a subject is notified of policy event occurrences, the subject delegates OE to interpret the triggered policy specifications. Policy interpretation consists in policy parsing, in controlling the dynamic conditions for policy applicability, in extracting the policy actions, and in activating binding/migration management operations accordingly, as described for CARMEN mobile proxies in the following. Let us note that OE implements a sequential policy enforcement: OE starts enforcing a new policy only after having completed all the management actions triggered by preceding events.

5.2 Context Manager

CM is responsible for dynamically establishing the context of any CARMEN client, thus determining its resource visibility. In particular, as more extensively described in Section 6, CARMEN mobile proxies cannot directly use the available resources and have to interrogate CM to obtain resource accessibility via a proper context object. In addition, system administrators can query CM to retrieve and modify the context objects of any CARMEN client at provision time.

To calculate the context object for a client, CM first merges the list of resources in the client access locality, obtained via the Discovery facility, and the list of globally available resources, retrieved via the Directory facility. Then, CM discards resources from the merged set depending on the current client location and the associated user/device profiles. For instance, if Alice's profile requires English-language local news resources, without considering the possibility of dynamic translation, noncompatible MNS service components are automatically eliminated from the context.

The obtained resource set corresponds to the user desiderata and the access device capabilities; it is subject to further restrictions and discarding due to access control policies to apply depending on the requesting user. After this access control filtering, the final result is a context object listing all the resources accessible to one client at the moment of the context determination request. CM represents a context object as a container of tuples, any tuple corresponding to an accessible resource and including a unique resource identifier, a resource descriptor, and information to properly manage the resource binding in case of mobility (see Section 6).

5.3 Event Manager

EM plays the crucial role of delivering the events relevant for triggering migration and binding policies. EM dispatches the registered events to interested policy subjects independently of subject migration during service provisioning, by exploiting the mobility-enabled naming facilities

available in the CARMEN middleware [20]. EM permits also to define aggregated events by composing several low-level monitoring indicators.

For instance, the CARMEN Monitoring facility is capable of sensing when a new portable device with IEEE 802.11b connectivity enters the area served by a Cisco Aironet 350 Access Point [27]. Similarly, CARMEN Monitoring exploits the Java Native Interface to integrate with the native location-tracking module of the Teleca distribution for Bluetooth access points [28]. In both cases, EM delivers the corresponding *DomainArrival* event to CM, which maintains the information about locally available resources to determine updated contexts, and to the other interested policy subjects, such as to CARMEN mobile proxies.

6 The CARMEN Shadow Proxies

The CARMEN middleware is centered on the distributed and dynamic deployment of context-aware mobile proxies over the fixed network to smooth the problems due to resource limits of wireless portable devices, to support operation asynchronicity/autonomy between client and service components, and to reduce the user/device connection time.

CARMEN provides any user, at the starting of her service session, with a personal mobile proxy, called *shadow proxy*, that migrates over the fixed network infrastructure to follow the user movements and that acts as the intermediary between the user wireless device and her context. Shadow proxies access any resource by passing through the context object, which is responsible for metadata-dependent management of resource bindings after the user movement.

Context awareness is crucial for CARMEN to dynamically choose the most suitable mobility strategy for the proxy in response to user movements and to apply the most suitable binding strategy after the proxy migration. For instance, in the MNS scenario, the visibility of Alice's location is necessary to trigger the migration of her proxy toward the current wireless access locality, or to simply deliver the MNS results to the new Alice's position without moving the proxy. After migration, the shadow proxy needs to obtain an updated context object with the bindings to the newly accessible MNS resources. In addition, binding policies can exploit the visibility of metadata to achieve dynamic service adaptation, e.g., by choosing among the available resources the one that better fits the user/device profiles. For instance, in the case of the micro mp3 reader used for accessing MNS, the binding policy can specify to connect either to a text-only based MNS component or to an HTML-to-text distiller, in its turn connected to an MNS component providing multimedia results.

We claim the suitability of the MA technology implements mobile shadow proxies for the wireless Internet. For this reason, we have built CARMEN on top of the Secure and Open Mobile Agent (SOMA) platform, which provides a wide range of mechanisms and tools to support secure and interoperable MAs for mobile computing [20]. More information and the SOMA platform code are available for download at http://lia.deis.unibo.it/Research/SOMA/. In particular, CARMEN exploits SOMA to implement shadow proxies as SOMA agents and to provide them with execution environments, called places, that offer the basic services for enabling MA communication and migration. Places typically model nodes and can be grouped into domains that correspond to network localities, e.g., Ethernet-based Local Area Networks with IEEE 802.11b/Bluetooth access points providing wireless connectivity to WiFi/Bluetooth portable devices. CARMEN middleware facilities are available in any CARMEN domain; shadow proxies usually run on places in the domain where the associated user and the corresponding wireless companion devices are currently connected.

CARMEN shadow proxies are application-independent middleware components that coordinate context management operations and binding reassessment on behalf of their responsible clients. CARMEN associates one shadow proxy for each user, with a 1-to-1 mapping. Shadow proxies usually follow their associated users in their movements among different SOMA domains, carry the reached service state and make possible to migrate service sessions dynamically. Other different mobility policies for shadow proxies can be specified and enforced, for instance, to support disconnected asynchronous operations, as described in the following section. It is the user reconnection at a new SOMA domain that triggers the shadow proxy migration. If the user does not reconnect before a timeout expiration, the associated proxy is automatically garbage-collected. The same holds if the new wireless access domain is unreachable due to network partitioning.

Shadow proxies retrieve the profile of characteristics of their companion devices and the profile of preferences of their users at their instantiation via the CARMEN Directory facility [20]. Let us note that the proxies need to interrogate the Directory only once, at the starting of the service session, being the profiles part of their state, which is maintained even after migration. Only the modification of the associated profiles triggers a corresponding event and a new profile request to the Directory.

To support dynamic binding management, shadow proxies are designed to refer, at start up, only to CM, without any direct access to resources. They request their contexts by passing profile information to the CM component in their current CARMEN domain. After context determination, CM returns back to the proxy a context object listing the identifiers of all accessible resources, either active or passive. At the beginning, all resources in the context are considered passive; a resource becomes active when the proxy asks CM to access and use that resource via a `getResource(ResourceID)` method.

For any active resource, the context object includes not only the resource identifier (which is the only information maintained for passive resources), but also the corresponding resource descriptor, the identifier of the binding strategy to apply upon the proxy arrival at a new domain, and a reference object that encapsulates the specific mechanism for implementing the associated binding strategy. The resource descriptor is an object with the same methods and constructor interface of the needed resource, returned back to the proxy by the `getResource()` invocation. CARMEN supports different mechanisms for reference objects, from Jini-based stubs for rebinding to RMI

stubs for remote reference; further details about CARMEN reference objects can be found in [16].

Afterward, shadow proxies can operate on active resources directly via the obtained resource descriptors. When a proxy arrives at a new locality, the local CM transparently updates the identifiers of the binding strategies for the context active resources to reflect possible changes in binding choices; if a binding strategy has changed, CM instantiates and adds new proper reference objects for the involved active resources in the context object.

Any context modification produces the notification to CM of a monitoring event with the data describing the change and, by default, of a context change event to the associated shadow proxy, which typically reacts by interrogating CM again to update its context object. Depending on service-specific requirements, the proxy can ask CM to transparently update its context with no need of explicit request, as it usually happens after a proxy migration to a new domain.

Shadow proxies interact with two types of additional middleware components to provide wireless portable devices with full service accessibility: device-specific clients and service adapters [19].

Device-specific clients are the only service components required to run on the portable devices. These clients announce the device entering/exiting into a CARMEN domain and exploit the responsible shadow proxies to send service requests and to receive service results. We have currently implemented three different types of lightweight device-specific clients. One runs on the KVM/CLDC/MIDP suite for Pocket PC handhelds with IEEE 802.11b connectivity. One is Bluetooth-based and written in C within the Ericsson Bluetooth Application and Training programming environment [29]. The third client is specific for Palm OS devices. Let us observe that, apart from the already described decoupling benefits, the choice of adopting shadow proxies over the fixed network and lightweight clients on the portable devices permits to exploit the MA-based CARMEN middleware also when providing services to limited devices that cannot host MA execution environments on them.

Service adapters are application-specific middleware components in charge of dynamically performing data transcoding over the fixed network. In particular, depending on the user/device profiles and the binding policies, CM can connect a shadow proxy either directly to the needed service component or to a suitable service adapter that filters service results from the service component before returning them back to the proxy. One shadow proxy can concurrently command several different adapters to carry on parallel service requests for the same user. Service adapters are implemented as SOMA agents that follow the movements of their shadow proxy. It is possible to specify various migration policies for the service adapters in response to the migration of the associated proxy, such as adapter immediate termination (the proxy will rebind to new adapters in the new destination domain) or adapter persistence in the locality until the end of the service session (to save processed service results on local persistent storage). For instance, the latter case can be useful in the case of adapter filtering operations on location-dependent information that the proxy is interested to collect asynchronously at the following user reconnection in that locality. The default migration choice, however, consists in automatic migrating all service adapters jointly with the shadow proxy they are working for.

7 The CARMEN-Based Mobile News Service

We have designed and implemented the context-dependent MNS, used as the running case study in the paper, on top of the CARMEN middleware. This section provides some design and implementation insights about MNS and shows the different CARMEN components at work during an actual service session. In addition, it exemplifies how CARMEN facilitates the realization of context-dependent services for the wireless Internet and reports some experimental results to quantitatively evaluate the effectiveness of the CARMEN middleware.

We have deployed MNS in a distributed environment consisting of several LANs with either IEEE 802.11b or Bluetooth access. Each LAN is modeled as a SOMA domain that hosts the CARMEN middleware facilities and provides news service components called "newspapers" representing local tabloids with general and district-specific information (local cinemas, events, restaurants, etc.). In addition, each domain provides execution environments for shadow proxies and service adapters of the MNS users currently in the locality.

In our example, Alice can access MNS by using her wireless access device equipped with a device-specific client, currently implemented for the most diffused access solutions (portable devices with either the J2ME/CLDC/MIDP suite and Wi-Fi connectivity, or PalmOS and Bluetooth support). The device-specific client allows Alice to subscribe to MNS, to specify news preferences, and to successively modify the provided profile information. In order to start her MNS service session, Alice must pass an authentication phase that associates her with both a unique user identifier and a unique device identifier corresponding to the currently used access terminal. The device identifier information is necessary for the CARMEN middleware to adapt service provisioning to the device profile, as illustrated in the following; user and device identifiers are cleanly separated to allow Alice to change her access device even during the same MNS session.

The device-specific client is the interface between Alice and her shadow proxy that CARMEN instantiates after her successful authentication. Fig. 3a shows an excerpt from the simple and reusable code of the `MNSProxy`. At its instantiation, the shadow proxy executes the `init()` method that retrieves the profiles of Alice and of her device from the CARMEN Directory facility, commands CM to determine the context object `myContext` by passing the user/device profile information, and invokes the `getResource("newspaper")` method on `myContext`. If the context object includes a resource called "newspaper," the invocation makes that resource active in the context, sets its binding strategy identifier to the "remote reference" default value, and returns back the `resourceID`

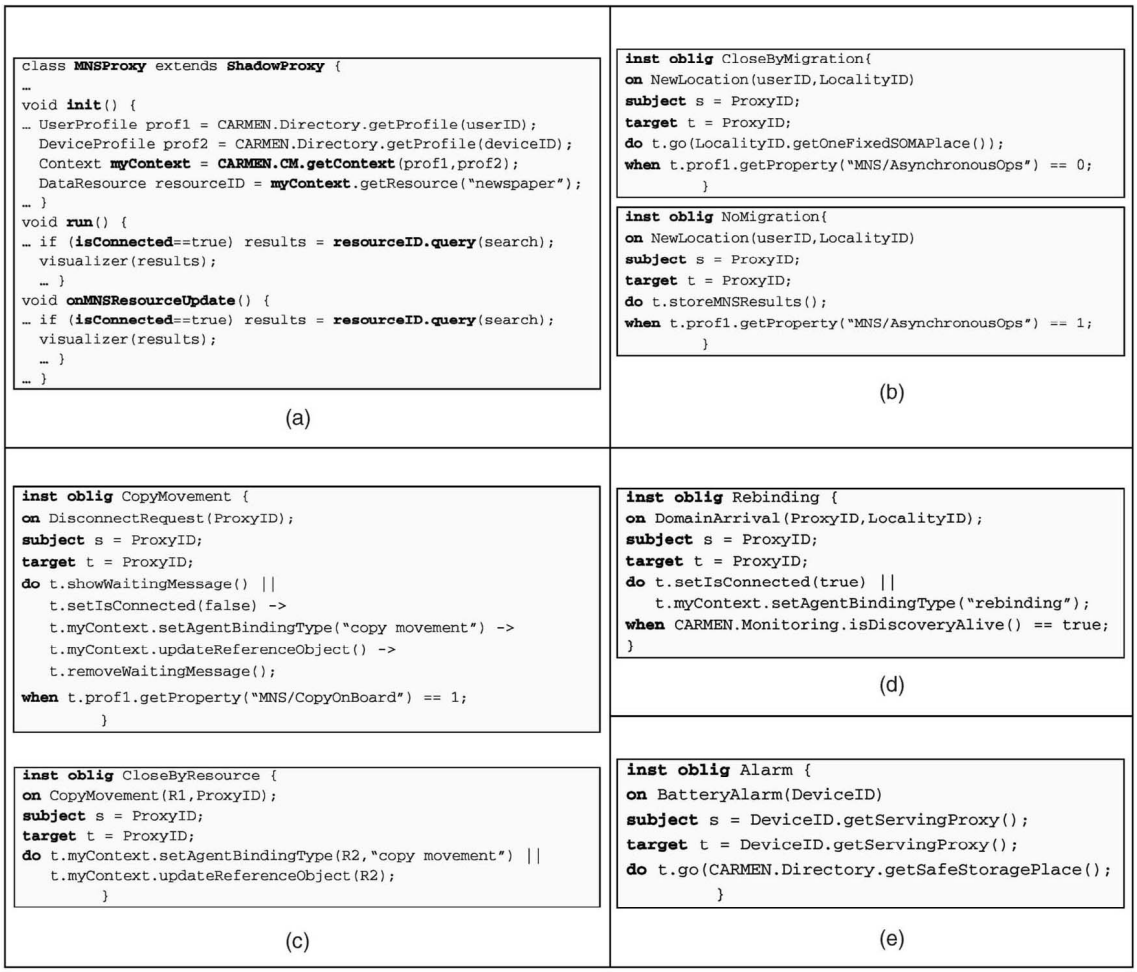

Fig. 3. (a) A code excerpt from MNSProxy, (b) migration policies for controlling proxy mobility, (c) a binding policy for disconnected operations together with the related colocality constraints, (d) a binding policy for automatically updating the MNS client context after change of domain, and (e) a migration policy update at service provision time.

resource descriptor to the proxy. After the initialization and after any migration to a new domain, the proxy executes its `run()` method: If the Alice's device is connected (`isConnected` set), the proxy forwards user-entered queries to `resourceID` and then invokes `visualizer()` to push the received results to the client. Fig. 3a shows that the update of an MNS resource triggers the same above described actions (`onMNSResourceUpdate()` method). Other proxy threads, not shown in the code excerpt, serve in the insertion of new queries and in maintaining the history of already browsed service results.

Without any modification of the `MNSProxy` implementation, it is possible to dynamically specify different migration/binding management policies to adapt MNS to work in different operating scenarios. Fig. 3b shows two examples of different migration policies to govern the shadow proxy migration in response to Alice's movement. When she connects to the new domain `LocalityID`, triggering the *NewLocation* event, the `CloseByMigration` policy commands her shadow proxy to migrate to a fixed host in the `LocalityID` domain and to execute there. In this case, the shadow proxy follows Alice's movements to maintain colocality. On the contrary, the enforcement of the `NoMigration` policy does not trigger any movement of the proxy, which continues to run in the domain of the previous Alice's connection, but produces the invocation of the proxy `storeMNSResults()` method. The method commands the proxy not to forward the received MNS results to the device-specific client on the companion device, but to redirect them to a disk file in the CARMEN place where the proxy is executing. This behavior is useful, for instance, to support the asynchronous collection of MNS results independently of Alice's connection/disconnection. Note that Alice can specify the `MNS/AsynchronousOps` attribute in her user profile to select between asynchronous/synchronous operations; the attribute is evaluated at runtime and determines the choice of the migration strategy to enforce.

CARMEN also supports the definition of different binding strategies, with no impact on the implementation of clients, service components, and shadow proxies. For instance, Alice can set the `MNS/CopyOnBoard` attribute in her profile if she desires to host an on-board copy of active resources on her portable device to continue reading news while disconnected. If Alice selects the disconnection menu option in the client interface (*DisconnectRequest* event), the `CopyMovement` policy in Fig. 3c commands the following set of actions. The proxy pops up a warning in the

companion device-specific client while the resource copy is in progress, to signal Alice when she can safely disconnect. To execute the copy process, the proxy `isConnected` state variable is set to false and the binding type is set to "copy movement" for any active resource in the proxy context object. Finally, the `updateReferenceObject()` method checks the binding strategy type for any active resource in the context, copies resources on board, and updates the reference objects accordingly.

In the case of copy movement of the newspaper resource R1, it could be convenient to also move other resource copies, e.g., R2, strictly related to R1, along with it. At the R1 copy movement, the CARMEN EM facility notifies a *CopyMovement* event and triggers the evaluation of the related `ClosebyResource` colocation policy, reported in Fig. 3c. `ClosebyResource` requests to move a copy of R2 anytime R1 (or one of its copies) migrates. If, when enforcing the `ClosebyResource` policy, the R2 profile specifies that R2 can be copied and transferred, a new entry for R2 is added in the `ProxyID` context object (`setAgentBindingType()` method) and the R2 copy movement is forced by the update of the corresponding reference object (`updateReferenceObject()` method).

CARMEN facilitates also the deployment of location-dependent MNS provisioning by simply specifying the `Rebinding` policy shown in Fig. 3d. When Alice enters a new domain of attachment followed by her shadow proxy (*DomainArrival* event), `Rebinding` sets `isConnected` and the binding strategy type to "rebinding" for any active resource in the ProxyID context object. The `Rebinding` policy actions are performed if and only if the CARMEN Discovery facility is working in the new domain of attachment. It is worth noting that if several instances of the same resource type are locally available, the user/device profiles drive the choice of which instances to bind. In fact, the reference object for the rebinding strategy performs simple parsing and processing operations on the attribute-value pairs contained in the profiles, and increments a score counter anytime a resource profile attribute is compatible with the corresponding user/device one. Then, the reference object chooses the resource with the maximum score and is updated with the dynamically downloaded Jini stub for that resource [16].

The flexibility and dynamic adaptability of MNS is even more evident in the case of policy modifications during service provisioning. CARMEN permits to disable old policies and to substitute them with new ones, or simply to add new policies to cope with previously unforeseen situations. CARMEN propagates policy changes to shadow proxies with no impact on their implementation. As an example, let us consider the case of extending the already deployed MNS with new operations to perform in case of battery shortage. The CARMEN MM facilitates the introduction of the new `Alarm` policy in Fig. 3e. After that, in case of `DeviceID` battery shortage (*BatteryAlarm* event), `Alarm` commands the associated proxy to migrate, together with the reached execution state, to a safe storage place retrieved via the Directory facility. Only after Alice's reconnection, possibly with a different device, the proxy will migrate again close to Alice to continue her MNS session.

7.1 MNS Experimental Evaluation

In addition to the flexibility and simplification of service implementation allowed by the adoption of metadata, a relevant factor that facilitates wireless Internet service development in CARMEN is code reutilization. In particular, the CARMEN framework provides two different possibilities of reuse, reuse of framework facilities and reuse of metadata, which we have measured in the MNS case to give some quantitative assessment of CARMEN effectiveness.

We have adopted the definition of Framework Facilities Reuse Level (FFRL) as the ratio between the size (in number of classes) of the code reused from the framework facilities and the size (in number of classes) of the overall service code (application-specific clients, servers, shadow proxies, and the whole framework) [30]. In the MNS case, FFRL results to be 67 percent. This FFRL value is achieved because MNS largely exploits both the CARMEN low-level facilities and the high-level ones, e.g., Directory for profile retrieval, event subscription/notification for obligation policies, Discovery in the Rebinding policy enforcement, and Monitoring in the Alarm policy enforcement.

Relevant state-of-the-art researches have specifically focused on evaluating the reuse level in framework-based application development, and have shown that, for frameworks of small/medium size, the reuse level tends to be settled around 80 percent, by considering the average of a wide set of applications of different nature [31]. Let us observe that our FFRL measurement, specific and limited to MNS, is lower than 80 percent mainly because CARMEN is sensibly larger in size than the framework considered in [31] and, therefore, CARMEN-based applications typically reuse a smaller subset of the framework facilities.

A CARMEN-specific reuse aspect relates to profile and policy metadata. All application-independent profiles represented in a standard format are highly reusable, e.g., CC/PP-compliant device profiles directly provided by device vendors (as the PalmOS one in Fig. 1). In addition, CARMEN supports the specification of policy templates that encode sets of common binding/migration strategies, from which policy instances can be created when needed. The adoption of templates in CARMEN promotes the reuse of existing binding/migration policies.

Given the very different representation formats and specification complexity of profiles/policies in CARMEN, we have decided to define and measure two different types of metadata reuse levels, the Profile Reuse Level (ProRL), and the Policy Reuse Level (PolRL). In fact, profiles typically consist of several hundreds of CC/PP, WSDL, or RDF lines of code, while policies are composed only by a dozen of Ponder lines on average. We have defined ProRL/PolRL as the ratio between the size (in number of lines) of the profile/policy specifications reused from the framework and the size (in number of lines) of the total number of profile/policy specifications required in an application. Let us also observe that the above ProRL definition could be considered arguable because different types of CARMEN profiles are expressed in different formats with different

expressive power; however, there are still no simple and widely accepted metrics for profiles.

The MNS prototype involves six device profiles, one user profile, 10 news resource profiles, and 18 different policies. Alice's profile, the news resource profiles, and the Alarm policy have been defined from scratch in MNS; the six device profiles were already available in CARMEN, as well as the policy templates from where the other 17 policies have been instantiated. In this specific case, ProRL is 70.6 percent and PolRL 78.7 percent, mainly because of the high reusability of device profiles and the large exploitation of existing policy templates.

Let us note that, not only the reuse level, but also the complexity of learning how to implement framework-based applications significantly impacts on development productivity, and represents a relevant factor in evaluating a framework from the software engineering point of view [31]. To reduce the framework learning effort, we have integrated CARMEN with a wide set of graphical tools to simplify metadata specification and reuse. These tools hide the complexity of profile/policy representation formats from developers by providing pre-defined schemas to fill in with the needed application-specific data, e.g., subjects, targets, and triggering events in the case of migration/binding policies [24].

The exploitation of a flexible context-aware framework for wireless Internet services, such as CARMEN, introduces different forms of runtime overhead, depending on the performance of the different framework functions involved, from monitoring to event distribution, from profile parsing to policy interpretation, from proxy migration to active resource reference update. During the testing of CARMEN-based MNS, we have conducted a number of measurements to give a quantitative estimation of the overhead introduced by several CARMEN components, also to verify the feasibility of our framework-based approach. Measurements have been taken on a 10Mbps Ethernet LAN of 1.7-GHz PentiumIV PCs. In particular, because metadata is the primary distinguishing feature of CARMEN, in the following we specifically focus on the experimental evaluation of the performance related to the adoption of metadata for proxy migration/binding.

For the sake of simplicity, here we present the experimental results for the `CloseByMigration`, `CopyMovement`, and `Rebinding` policies, by comparing the migration and binding/rebinding costs for two different shadow proxy implementations: the proxy version of Section 7 with migration and binding/rebinding strategies separately specified in terms of policies (`MNSProxy`), and an alternative implementation of a proxy with the same behavior but with migration and binding/rebinding strategies directly hard-coded into its code (`HardCodedMNSProxy`). The bytecode size of `HardCodedMNSProxy` results to be 93.6KB while the `MNSProxy` size, with all the 18 needed policies on-board, is 99.2KB. In addition, in the testing deployment scenario, the proxies have two profiles (the Alice's profile and the PalmOS device one) included in their state and two active news resources in their context objects, with each resource size of 100KB.

TABLE 1
Average Response Times for `MNSProxy` and `HardCodedMNSProxy`

	Response Time (ms)		
	CloseByMigration	CopyMovement	Rebinding
HardCodedMNSProxy	386	453	36
MNSProxy	415	488	61

We have decided to measure the response time, defined as the time interval between an event notification received at the shadow proxy and the completion of the consequent management action of migration and binding/rebinding. Table 1 reports the average response time for `MNSProxy` and `HardCodedMNSProxy`. For instance, the `HardCodedMNSProxy CloseByMigration` cell reports the costs for parsing the user profile and for moving the proxy between two places (mainly, the time to establish a connection between the origin and the destination place, and to serialize/deserialize the proxy). The same strategy driven by metadata exhibits a slightly higher response time due to the additional time spent, at event notification, for selecting the triggered policy and parsing it to extract constraints and actions. Similar considerations apply to the other strategies, `CopyMovement` and `Rebinding` (note that the `Rebinding` cases do not involve any proxy/resource serialization/deserialization). In general, as expected, migration and binding/rebinding actions introduce a slightly larger overhead when driven by metadata. However, we claim that this difference is counterbalanced by the augmented flexibility and reusability.

Finally, we have also evaluated the performance of an alternative `MNSProxy` version that does not carry the migration/binding policies on-board, but retrieves them when needed by exploiting the CARMEN Directory. In this case, the response time also includes additional factors due to Directory interrogation, which depends on several deployment choices and CARMEN-independent aspects, such as the degree of replication/partitioning of the directory components, the directory size and the current load. For this reason, we have decided to report in Table 1 the response times for `MNSProxy` and `HardCodedMNSProxy` because they best point out the CARMEN costs associated with metadata-specific management.

8 Related Work

Several research efforts have addressed the general issue of middleware solutions to support user/device mobility, by facing very different aspects, from the provisioning of virtual home environments to roaming users in 3G communications, to the effective synchronization of data replicas on mobile devices, and to profile-based content tailoring/adaptation [7], [32], [33], [34]. It is relevant to observe that, notwithstanding the very different issues addressed, most solutions propose the adoption of some forms of metadata to drive the service/middleware behavior at runtime. For instance, industry-driven initiatives such as the Synchronous Markup Language (SyncML) and the Synchronized Multimedia Integration Language (SMIL) propose metadata, respectively, to

maintain replica modification flags and to describe multimedia presentations with alternative contents [35], [36]. We do not intend to provide here a general survey of the state-of-the-art middleware for dynamic service management in mobile scenarios, but only to focus on the research that explicitly deals with the resource management aspects specifically addressed by CARMEN, i.e., flexible context-dependent management of binding/mobility strategies at provision time.

By focusing on mobility strategies, few solutions have been recently proposed for dynamically deciding when, where, and which software components to allocate in order to adapt to possible variations in the execution environment. All proposals follow the key design choice of separating mobility and computational concerns. The approach in [37] focuses on MA-based applications and suggests the separation of MA applications into three aspects: the functional, the mobility, and the management ones, by suggesting a separate programming of the three parts. Each MA is associated with an array of three elements: The first contains the MA code, the second includes all the data referred by the MA, and the third describes the MA path of successive execution environments to visit. Another interesting approach is FarGo, which supports the specification of policies driving the runtime allocation of mobile software components [38]. However, in the current FarGo implementation, allocation policies are encoded within the application by using a specific Application Programming Interface. The exploitation of an event-based scripting language for specifying allocation policies separately from the application logic is still under development. The MAGE project, instead, introduces the mobility attribute programming abstraction to describe the mobility semantics of application components. Programmers can associate components with mobility attributes to control their dynamic allocation [39]. However, MAGE leaves to the programmer the burden of manually implementing the proper binding between application components and needed resources, depending on the specified mobility attributes. The DACIA framework for developing and deploying reconfigurable mobile applications represents another interesting research activity [40]. DACIA provides mechanisms to dynamically alter the rules for application reconfiguration. However, it does not integrate with any high-level language for the specification of service reconfiguration policies clearly separated from the service logic. A relevant proposal at a very preliminary stage is the RAM infrastructure, suggesting to exploit reflection for the mobility management of object clusters, each one associated with a metaobject governing its mobility behavior [41]. RAM adopts a reflective language that supports the linking of metaobjects to the application logic at compile time and their dynamic modification.

Reflection represents an interesting design guideline to achieve context awareness in middleware solutions, but it is difficult to integrate with legacy systems typically implemented by nonreflective programming languages. On the contrary, policy-based approaches, as the CARMEN one, require the availability of monitoring and event facilities to trigger the policy enforcement anytime relevant context changes occur, but can apply also to legacy services, independently of their implementation language.

By focusing on flexible binding management, the design principle of separation of concerns has ruled the design of a few recent proposals in the literature. They differ on how to achieve the separation between binding concerns and application logic. The approach in [42] uses reflection to define customizable binding strategies implemented as basic reusable metaobjects attached to any mobile application component. However, the linking between application components and binding strategies is performed at the beginning of the execution, and cannot change at provision time without an execution restart. Another interesting approach is the FarGo system (already presented above), which supports the programming of different binding relationships as a separate component of the application code [38]. However, similar to [42], the binding strategies are decided and associated to FarGo entities only at application loading. It is worth noting that the CARMEN support for mobility/binding management has different points in common with the above approaches; the main distinctive point is the CARMEN possibility to specify migration/binding strategies in terms of high-level obligation policies and to modify them even during service provisioning, without any impact on service implementation.

Finally, several research activities are exploiting the idea of proxies to smooth the heterogeneity/discontinuities in available resources between the fixed Internet and the wireless access environment. For instance, in [43], [44] middleware components are statically placed at the wired-wireless edges to perform local monitoring of the offered quality and SMIL-based content adaptation in case of multimedia streaming.

Due to the novelty of the MA technology, few researches have proposed middleware solutions that exploit MAs to implement mobile proxies. The ACTS OnTheMove project has adapted an existing MA platform with a Mobile Application Support Environment to provide a statically installed proxy to support laptop mobility between fixed and wireless networks [45]. Dartmouth Agent TCL provides a docking station abstraction (a sort of fixed proxy available in any domain) in charge of forwarding MAs/messages to mobile laptops independently of their current location [46]. Other MA proposals mainly concentrate on proxy solutions for profile-based realization of virtual home environments [47]. To our knowledge, apart from our approach, Grasshopper is the only MA platform addressing the specific issues of limited portable devices that access wireless Internet services. However, Grasshopper focuses on providing a lightweight version of the MA execution environment that can be suitably accommodated in access devices with either Personal Java or the Java 2 Micro Edition [48]. The CARMEN middleware, instead, is original in addressing the issues of Web service accessibility from wireless portable devices without any version of the JVM, by adopting MA-based mobile proxies working in the fixed network.

9 Conclusions and On-Going Work

The development and deployment of wireless Internet services motivate flexible and mobile middleware solutions with full context visibility and capable of properly handling context modifications during service provisioning. In addition, the complexity of the wireless Internet scenario suggests a clear separation of concerns between resource binding/mobility strategies and service logic implementation, to achieve the requested level of dynamicity, flexibility, and reusability of both middleware and service components. Novel programmable middleware solutions, integrated with different types of high-level metadata, can provide the needed management configurability while hiding low-level mechanisms and implementation details from service developers and system administrators.

First, experiences with CARMEN have shown that our middleware can simplify service design and implementation, can provide effective service reconfiguration in response to runtime context changes, and can favor component reusability in different deployment conditions. These encouraging results are stimulating further research to extend the current middleware prototype and to develop other mobile services on top of it. In particular, we are working on the integration of the CARMEN middleware with additional multimedia-specific service adapters, which we have implemented for the dynamic QoS adaptation (filtering, downscaling, transcoding, etc.) of video-on-demand flows in the context of other research projects [49].

Acknowledgments

This work was supported by the Italian Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) in the framework of the FIRB WEB-MINDS Project "Wide-scale Broadband Middleware for Network Distributed Services" and by the Italian Consiglio Nazionale delle Ricerche (CNR) in the framework of the Strategic IS-MANET Project "Middleware Support for Mobile Ad-hoc Networks and their Application."

References

[1] W. Stallings, *Wireless Communications and Networks.* Pearson Education, Aug. 2001.
[2] S. Agarwal, A. Agrawala, S. Banerjee, T. Bao, K. Kamel, A. Kochut, C. Kommareddy, R.L. Larsen, T. Nadeem, P. Thakkar, A. Udaya Shankar, A. Youssef, and M. Youssef, "Rover: Scalable Location-aware Computing," *Computer,* vol. 35, no. 10, pp. 46-53, Oct. 2002.
[3] M.S. Corson, J.P. Macker, and V.D. Park, "Mobile and Wireless Internet Services: Putting the Pieces Together," *IEEE Comm. Magazine,* vol. 39, no. 6, pp. 148-155, June 2001.
[4] *IEEE Wireless Comm.,* Special Section on Context-aware Pervasive Computing, G.D. Abowd, M.R. Ebling, H.-W. Gellersen, H. Lei, G. Hunt, eds., vol. 9, no. 5, Oct. 2002.
[5] A. Fuggetta, G.P. Picco, and G. Vigna, "Understanding Code Mobility," *IEEE Trans. Software Eng.,* vol. 24, no. 5, pp. 342-361, May 1998.
[6] L. Bos and S. Leroy, "Toward an All-IP-based UMTS System Architecture," *IEEE Network,* vol. 15, no. 1, pp. 36-45, Jan.-Feb. 2001.
[7] G.-C. Roman, G.P. Picco, and A.L. Murphy, "Software Engineering for Mobility: A Roadmap," *Proc. 22nd Int'l Conf. Software Eng. (ICSE '00),* pp. 241-258, June 2000.
[8] H. Huber, M. Jarke, M.A. Jeusfeld, H.W. Nissen, and G.V. Zemanek, "Managing Multiple Requirements Perspectives with Metamodels," *IEEE Software,* vol. 13, no. 2, pp. 37-48, Mar. 1996.
[9] J. Moffett and M. Sloman, "Policy Hierarchies for Distributed Systems Management," *IEEE J. Selected Areas in Comm.,* vol. 11, no. 9, pp. 1404-1414, Dec. 1993.
[10] S. Avancha, T. Finin, and A. Joshi, "Enhanced Service Discovery in Bluetooth," *Computer,* vol. 35, no. 6, pp. 96-99, June 2002.
[11] IETF SVRLOC Working Group—Service Location Protocol (SLP), http://www.srvloc.org, 2003.
[12] K. Arnold, R. Scheifler, J. Waldo, B. O'Sullivan, and A. Wollrath, *Jini Specification.* Addison-Wesley, June 1999.
[13] F. Curbera, M. Duftler, R. Khalaf, N. Mukhi, W. Nagy, and S. Weerawarana, "Unraveling the Web Services: an Introduction to SOAP, WSDL, and UDDI," *IEEE Internet Computing,* vol. 6, no. 2, pp. 86-93, Mar.-Apr. 2002.
[14] R. Riggs, J. Waldo, and A. Wollrath, "Java-centric Distributed Computing," *IEEE Micro,* vol. 17, no. 3, pp. 44-53, May-June 1997.
[15] J. Snell, K. MacLeod, D. Tidwell, and P. Kulchenko, *Programming Web Services with SOAP.* O'Reilly & Assoc., Dec. 2001.
[16] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli, "Policy-driven Binding to Information Resources in Mobility-enabled Scenarios," *Proc. Fourth Int'l Conf. Mobile Data Management (MDM '03),* Jan. 2003.
[17] *IEEE Comm. Magazine,* Special Section on Mobile Agents, A. Karmouch, ed., vol. 36, no. 7, July 1998.
[18] F. Eliassen, A. Andersen, G.S. Blair, F. Costa, G. Coulson, V. Goebel, Ø. Hansen, T. Kristensen, T. Plagemann, H.O. Rafaelsen, K.B. Saikoski, and W. Yu, "Next Generation Middleware: Requirements, Architecture and Prototypes," *Proc. Seventh IEEE Workshop Future Trends in Distributed Computing Systems (FTDCS '99),* pp. 60-65, 1999.
[19] P. Bellavista, A. Corradi, and C. Stefanelli, "The Ubiquitous Provisioning of Internet Services to Portable Devices," *IEEE Pervasive Computing,* vol. 1, no. 3, pp. 81-87, July-Sept. 2002.
[20] P. Bellavista, A. Corradi, and C. Stefanelli, "Mobile Agent Middleware for Mobile Computing," *Computer,* vol. 34, no. 3, pp. 73-81, Mar. 2001.
[21] W3 Consortium—Composite Capability/Preference Profiles (CC/PP), http://www.w3.org/Mobile, 2003.
[22] S. Decker, P. Mitra, and S. Melnik, "Framework for the Semantic Web: an RDF Tutorial," *IEEE Internet Computing,* vol. 4, no. 6, pp. 68-73, Nov.-Dec. 2000.
[23] C.E. Ortiz and E. Giguere, *Mobile Information Device Profile for Java 2 Micro Edition (J2ME): Professional Developer's Guide.* Wiley, Dec. 2001.
[24] Imperial College—Ponder, http://www-dse.doc.ic.ac.uk/Research/policies/ponder.shtml, 2003.
[25] R. Montanari, C. Stefanelli, and N. Dulay, "Flexible Security Policies for Mobile Agent Systems," *Microprocessors and Microsystems,* vol. 25, no. 2, pp. 93-99, May 2001.
[26] P. Bellavista, A. Corradi, and C. Stefanelli, "Java for On-line Distributed Monitoring of Heterogeneous Systems and Services," *The Computer Journal,* vol. 45, no. 6, pp. 595-607, Nov. 2002.
[27] Cisco Systems—Cisco Aironet 350 Series, http://www.cisco.com/en/US/products/hw/wireless/ps458/index.html, 2003.
[28] Teleca AB—Bluetooth Local Infotainment Point (BLIP), http://www.teleca.com, 2003.
[29] Ericsson—Bluetooth Application and Training Tool Kit, http://www.ericsson.com/bluetooth, 2003.
[30] J.S. Poulin, *Measuring Software Reuse: Principles, Practice and Economic Models.* Addison Wesley, 1997.
[31] M. Morisio, D. Romano, and I. Stamelos, "Quality, Productivity, and Learning in Framework-Based Development: an Explanatory Case Study," *IEEE Trans. Software Eng.,* vol. 28, no. 9, pp. 876-888, Sept. 2002.
[32] C. Mascolo, L. Capra, and W. Emmerich, "Middleware for Mobile Computing," *Networking 2002 Tutorial Papers,* LNCS 2497, pp. 20-58, Nov. 2002.
[33] N. Davies and H.-W. Gellersen, "Beyond Prototypes: Challenges in Deploying Ubiquitous Systems," *IEEE Pervasive Computing,* vol. 1, no. 1, pp. 26-35, Jan.-Mar. 2002.
[34] J.A. Moura, J.M. Oliveira, E. Carrapatoso, and R. Roque, "Service Provision and Resource Discovery in the VESPER VHE," *Proc. IEEE Int'l Conf. Comm. (ICC '02),* Apr. 2002.
[35] S. Agarwal, D. Starobinski, and A. Trachtenberg, "On the Scalability of Data Synchronization Protocols for PDAs and Mobile Devices," *IEEE Network,* vol. 16, no. 4, pp. 22-28, July-Aug. 2002.

[36] D.C.A. Bulterman, "SMIL 2.0: Examples and Comparisons," *IEEE Multimedia,* vol. 9, no. 1, pp. 74-84, Jan.-Mar. 2002.
[37] D. Johansen, K.J. Lauvset, and K. Marzullo, "Factoring Mobile Agents," *Proc. IEEE Int'l Conf. Eng. Computer-Based Systems (ECBS '02),* pp. 253-257, 2002.
[38] O. Holder, I. Ben-Shaul, and H. Gazit, "Dynamic Layout of Distributed Applications in FarGo," *Proc. 21st Int'l Conf. Software Eng. (ICSE '99),* pp. 163-173, 1999.
[39] E. Barr, M. Huangs, and R. Pandey, "MAGE: a Distributed Programming Model," *Proc. 21st IEEE Int'l Conf. Distributed Computing Systems (ICDCS '01),* pp. 303-312, 2001.
[40] R. Litiu and A. Prakash, "DACIA: A Mobile Component Framework for Building Adaptive Distributed Applications," *Proc. Int'l Middleware Symp. Principles of Distributed Computing (PODC '00),* July 2000.
[41] T. Ledoux and N.M.N. Bouraqadi-Saâdani, "Adaptability in Mobile Agent Systems using Reflection," *Proc. Int'l Workshop Reflective Middleware in Middleware '00,* Apr. 2000.
[42] E. Tanter and J. Piquer, "Managing References upon Object Migration: Applying Separation of Concerns," *Proc. 21st Int'l Conf. Chilean Computer Science Society (SCCC '01),* pp. 264-272, 2001.
[43] T. Yoshimura, T. Ohya, T. Kawahara, and M. Etoh, "Rate and Robustness Control with RTP Monitoring Agent for Mobile Multimedia Streaming," *Proc. IEEE Int'l Conf. Comm. (ICC '02),* Apr. 2002.
[44] T. Yoshimura, Y. Yonemoto, T. Ohya, M. Etoh, and S. Wee, "Mobile Streaming Media CDN Enabled by Dynamic SMIL," *Proc. 11th Int'l World Wide Web Conf. (WWW '02),* May 2002.
[45] E. Kovacs, K. Rohrle, and M. Reich, "Integrating Mobile Agents into the Mobile Middleware," *Proc. Mobile Agents Int'l Workshop (MA '98),* 1998.
[46] S. Chawla, G. Cybenko, R. Gray, D. Kotz, S. Nog, and D. Rus, "Agent TCL: Targeting the Needs of Mobile Computers," *IEEE Internet Computing,* vol. 1, no. 4, pp. 58-67, July 1997.
[47] S. Lipperts and A. Park, "An Agent-based Middleware: a Solution for Terminal and User Mobility," *Computer Networks,* vol. 31, pp. 2053-2062, Sept. 1999.
[48] IKV++ Technologies AG—enago mobile, http://www.ikv.de, 2003.
[49] P. Bellavista and A. Corradi, "Active Middleware for Internet Video on Demand: the QoS-aware Routing Solution in ubiQoS," *Elsevier Microprocessors and Microsystems,* vol. 27, no. 2, pp. 73-83, Mar. 2003.

Paolo Bellavista received the Laurea degree in electronics engineering and the PhD degree in computer engineering from the University of Bologna. He is currently a research associate of computer engineering, and is with the Department of Electronics, Computer Science, and Systems (DEIS), University of Bologna. His research activities span from mobile agents and pervasive computing to systems/service management, location/context-aware services, and adaptive multimedia. He is member of the IEEE, the IEEE Computer Society, the ACM, and the Italian Association for Computing. His Web page is at http://lia.deis.unibo.it/Staff/PaoloBellavista/.

Antonio Corradi received the Laurea degree in electronics engineering from the University of Bologna, and the MS degree in electrical engineering from Cornell University. He is currently a full professor of computer engineering, and is with the Department of Electronics, Computer Science, and Systems (DEIS), University of Bologna. His research interests include distributed systems, object and agent systems, network management, and distributed and parallel architectures. He is a member of the IEEE, the IEEE Computer Society, the ACM, and the Italian Association for Computing. His Web page is at http://lia.deis.unibo.it/Staff/AntonioCorradi/.

Rebecca Montanari received the Laurea degree in electronics engineering and the PhD degree in computer engineering from the University of Bologna. She is currently a research associate of computer engineering, and is with the Department of Electronics, Computer Science, and Systems (DEIS), University of Bologna. Her research primarily focuses on policy-based networking and systems/service management, mobile agent systems, security management mechanisms, and tools in both traditional and mobile systems. She is a member of the IEEE, the IEEE Computer Society, and the Italian Association for Computing. Her Web page is at http://lia.deis.unibo.it/Staff/RebeccaMontanari/.

Cesare Stefanelli received the Laurea degree in electronics engineering and the PhD degree in computer engineering from the University of Bologna. He is currently an associate professor of computer engineering in the Department of Engineering at the University of Ferrara. His research interests include distributed and mobile computing, mobile code, network and systems management, and network security. He is a member of the IEEE, the IEEE Computer Society, and the Italian Association for Computing. His Web page is at http://www.ing.unife.it/docenti/CesareStefanelli/.