Mobile Middleware Solutions for the Adaptive Management of Multimedia QoS to Wireless Portable Devices

Paolo Bellavista, Antonio Corradi
Dip. Elettronica, Informatica e Sistemistica - Università di Bologna
Viale Risorgimento, 2 - 40136 Bologna - ITALY
Phone: +39-051-2093001; Fax: +39-051-2093073
{pbellavista, acorradi}@ deis.unibo.it

Abstract

New challenging service scenarios are integrating wireless portable devices with limited and heterogeneous capabilities. They are expected to access both traditional and novel (context-dependent) Internet services. This not only calls for novel infrastructures to support the integration of mobile devices with the fixed network, but also stresses the necessity of negotiation-time tailoring and provision-time adaptation of Quality of Service (QoS). The paper presents a flexible and dynamic middleware for the management of multimedia QoS to wireless devices that can roam during service provisioning. The middleware exploits mobile agents that act as device shadow proxies over the fixed network to transparently follow the device movements between wireless localities, to negotiate the proper QoS level, and to dynamically adapt multimedia flows depending on device profiles and user preferences. In particular, the paper focuses on how the middleware achieves the on-line visibility of the device change of locality in a portable way over different implementations of different wireless technologies, i.e., IEEE 802.11 and Bluetooth. The first experimental results show that, notwithstanding the application-level approach, the middleware reorganization time is compatible with the requirements imposed by the addressed multimedia scenario.

1. Introduction

Recent advances in wireless networking and the enlarging market of wireless-enabled portable devices stimulate the provisioning of distributed services to a wide set of client terminals with very heterogeneous and limited resources. Service providers and wireless network operators have to face new challenging and state-of-the-art technical issues, toward both the deployment of distributed applications for mobile ad-hoc networks and the full seamless integration of wireless client devices with the traditional fixed Internet. The first scenario is just moving its first steps and still investigating solutions mostly at the network level, e.g., for multi-hop cooperative routing [1]. At the opposite, the second scenario, which we will call *wireless Internet* in the following, already starts to exhibit research and commercial network-level solutions to support mobile connectivity [2, 3]. However, the investigation of several aspects of service accessibility in the global and open wireless Internet, such as dynamic un/installation of infrastructure/service components, configuration management, service content adaptation, security, and interoperability, is still at its beginning and calls for flexible state-of-the-art middleware solutions at the application level [4, 5].

In addition, the wireless Internet stimulates research on several novel classes of services (and on methodologies for their development) where service results and the offered Quality of Service (QoS) levels may depend on the provisioning *context*, i.e., the logical set of resources that a client can access due to some properties of the provisioning environment, such as client location, access device capabilities, user preferences, and mutual relationships with currently local users/terminals [6].

In particular, the provisioning of context-dependent services for the wireless Internet must take into consideration the specific constraints of access portable devices, i.e., their limits on local resources and their high heterogeneity. On the one hand, limited processing power, memory and file system make portable devices unsuitable for traditional services designed for the fixed network and require downscaling to fit less powerful access terminals. On the other hand, portable devices exhibit heterogeneity of hardware capabilities, installed operating systems, supported software and network technologies. Heterogeneity makes impossible to provide all needed service versions with the suitable and statically tailored QoS levels. Middleware solu-

Proceedings of the Ninth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'03)
0-7695-2054-5/04 $20.00 © 2004 **IEEE**

tions should tailor the provided QoS to the specific characteristics of the client access device at negotiation time and possibly adapt the QoS to context changes at provision time. Moreover, the client limited memory suggests the dynamic deployment of middleware components over the fixed network, when and where needed, by following the roaming of client devices in different wireless localities during the service session, while portable devices could host only thin clients, loaded by need and automatically discarded after service.

The paper proposes a middleware, called Mobile UbiQoS (MU), that can extend the existing Internet infrastructure to support the distribution of Video on Demand (VoD) to wireless portable devices with strict limits on local resources. The main design idea is to dynamically deploy MU components that act on behalf of wireless devices over fixed hosts in the network localities currently hosting the devices. MU middleware components can operate autonomously to carry on service requests even in case of temporary device disconnection, can follow device movements in the different wireless network localities, and can operate on VoD flows (downscaling the offered QoS level) to fit the device visualization capabilities.

MU components are based on the Mobile Agent (MA) technology. MAs are active entities that can migrate from one network node to one another during their execution by carrying their code while preserving their reached execution state. Active components can exploit the MA properties of mobility, asynchronicity and autonomy during service provisioning. The MA technology enables the deployment of the MU middleware by enhancing the fixed network and by extending infrastructure services only when and where needed, without imposing any operation suspension [7].

MU is the result of the extension to wireless portable devices of the existing ubiQoS middleware for the QoS tailoring, control and adaptation of VoD flows over best-effort global networks [8]. In particular, the paper focuses on a very crucial aspect of the MU middleware: how MU senses the change of wireless network locality of its (possibly heterogeneous) access devices, by achieving application-level on-line visibility of client location in a completely portable way over different implementations of different wireless technologies (IEEE 802.11 on Microsoft Windows and Linux, Bluetooth on top of the Java Virtual Machine - JVM). On the one hand, MU location visibility enables the application-transparent integration of traditional Internet VoD services, which may become location-dependent based on the operations of the MU context-aware middleware, with no impact on the VoD application logic. On the other hand, MU provides portable mechanisms to give direct location visibility mechanisms to give direct location visibility to VoD clients when interested in developing novel location-aware service clients. Other details about the MU architecture and implementation can be found in [9].

First experimental results show that the MU middleware reorganization time in response to a client device change of wireless locality is compatible with the challenging requirements imposed by the addressed area of VoD distribution, notwithstanding the MU application-level approach. In addition, these encouraging results are stimulating further research to estimate wireless device trajectories and to predict the next visited locality, in order to migrate in advance the MU middleware components with enough buffered VoD content to support continuous VoD streaming independently of the client roaming.

The paper is organized as follows. Section 2 presents the main motivations for the adoption of the MA technology in the QoS management of service provisioning to wireless portable devices. Section 3 presents the architecture and main functions of the MU components, while Section 4 focuses on the portable implementation of the MU solution for giving location visibility to either the MU middleware components or the location-aware VoD clients. Section 5 shows our middleware in action by sketching the case study of a movie-info service built on top of MU. Conclusions and on-going work end the paper.

2. Mobile Agents for Service QoS Management to Wireless Portable Devices

Most recent research activities recognize the necessity of novel programming paradigms and technologies that overcome the traditional limits of the client/server model for mobile and pervasive computing scenarios. In particular, the interest in mobile code technologies is growing, and, more specifically, the MA technology seems very suitable for the design and implementation of mobile computing middlewares because MA requirements coincide with mobility ones [10, 11]. The MA applicability for mobile computing is recognized also in the telecommunications domain. In fact, the Telecommunications Information Networking Architecture (TINA) consortium has promoted the specification of novel middlewares for personal mobility that integrate TINA access sessions with MAs [12].

In particular, we claim the MA technology as a promising solution for QoS management over the best-effort wireless Internet. In fact, it allows developing and dynamically deploying decentralized infrastructures of proxies that work on behalf of portable devices. These proxies can execute over fixed hosts in the network localities close to the current device allocation to support wireless connectivity to the fixed

Internet. MA-based proxies can follow device movements during service provisioning by maintaining the session state and can install automatically in any newly visited network locality, only when needed. In addition, the MA adoption facilitates the achievement of crucial properties, such as dynamicity, autonomy in QoS tailoring/adaptation, context awareness, and asynchronicity described in the following.

Middleware solutions for the wireless Internet should provide dynamicity, intended as the possibility to modify and extend the fixed network infrastructure with new components and protocols to support device accessibility and to adapt the offered QoS levels to evolving client requirements, even during service provisioning. Dynamic installation and code distribution typical of MAs are crucial when dealing with portable devices with limited and highly heterogeneous hardware/software characteristics.

MAs can effectively tailor the QoS level of provided services to user requirements and resource availability at negotiation time: dedicated MAs can retrieve user/device profile information, can propagate this information to current access points and customize the QoS of service flows depending on the current access devices and the already admitted service sessions. In addition, MAs can simplify the QoS adaptation in response to modifications in the availability of network resources at provision time [9]. For instance, MAs can dynamically migrate where needed to monitor network resources locally; this visibility of local monitoring indicators permits to trigger management operations to correct the actually supplied QoS (re-negotiation, format transcoding, additional communication channels for VoD distribution, …) by exploiting locality to congested resources.

Middleware solutions for the wireless Internet should also be capable of providing context visibility to application developers, when needed, to enable the design and implementation of context-aware services. Context awareness (and in particular location visibility) are typical of the MA programming paradigm, where location awareness is required to drive MA mobility decisions and is propagated up to the application level to support service-specific management choices depending on dynamic conditions.

Finally, pervasive computing scenarios also take advantage of asynchronicity between requests of user/device operations and their execution. For instance, wireless connections impose strict constraints on available bandwidth and on communication reliability, and force to minimize the portable device connection time. The MA paradigm does not assume continuous network connectivity and expects connections to last only for the time needed to inject agents from wireless devices to the fixed network. MAs are autonomous and permit to carry on services even after the disconnection of the launching users/devices with the mission of giving service results back at their reconnection [10, 11].

Let us note that MA platforms have investigated and proposed solutions to achieve other relevant middleware properties, such as security and interoperability. Even if not specific of the wireless Internet domain, the availability of these solutions is important and can significantly leverage the adoption of MA-based middlewares for pervasive computing [9, 13]. Due to the novelty of the MA technology, however, there are not many MA environments already employed to implement middleware solutions for the wireless Internet. For a rapid overview of the most relevant recent approaches that propose MA-based partial solutions for pervasive computing, please refer to [5].

3. Mobile UbiQoS: Architecture and Middleware Components

The provisioning of wireless Internet services to portable devices usually requires downsizing service contents to suit the specific limits of access terminals. In particular, dynamic content negotiation and tailoring are crucial for resource-consuming services such as VoD distribution. In addition, device mobility requires several other support operations that may be too expensive to be done by severely limited devices on their own, e.g., local/global resource finding, binding and adjustment. On the one hand, local discovery operations may consume a large amount of resources in the environment exploration and in negotiations with available entities and services. On the other hand, even global identification and retrieval of user-related properties, such as profiles of user preferences, profiles of access device capabilities, and security certificates, from directory-based name services can be too long to be directly controlled and managed by terminals with severe resource constraints [5].

By focusing on where to operate VoD tailoring depending on the access device visualization capabilities and the installed software, typically by reducing frame rate/resolution and format transcoding, some first solutions for dynamic tailoring are emerging. They typically limit the choice to perform the QoS tailoring of the VoD flow at either the server or the client. In server-based tailoring solutions, the service provider is in charge of either selecting the version of the requested VoD flow with the most suitable QoS level among a pool of statically pre-determined ones (off-line tailoring) or dynamically operating downscale transformations on the unique high-quality stored version (on-the-fly tailoring). Let us additionally note that

the server-based approach makes distributed caching infrastructures less effective because VoD flows are tailored since the beginning to fit the specific client requirements [9]. On the opposite, client-based tailoring decentralizes the different charges: it is the client site that downscales the VoD flow before visualizing it, with a significant engagement of client computational resources. Consequently, the client-based approach is usually not applicable to portable devices.

VoD service provisioning in the wireless Internet can significantly benefit from distributed and active infrastructures that are hosted in the fixed network and consist of mobile middleware proxies working on behalf of portable devices. Proxies can dynamically distribute over the paths connecting server and clients to locally operate on VoD flows. They can decide the best QoS management operations to perform and can act as distributed cache repositories of previously requested flows in successive service requests. In addition, these proxies can become in charge of additional management operations, such as supporting connectivity and discovering the needed resources/service components, either in the current device locality or in the whole Internet environment. Mobile proxies could also follow device movements during service provisioning and install automatically, only when needed, depending on the fixed network localities visited by portable devices.

The importance of proxies that move close to clients is recognized also in the Jini discovery solution where proxy objects are dynamically installed at the JVM on the client side to handle the interaction with the discovered remote services. However, most current portable devices do not host any version of the JVM; even in case of devices supporting limited versions of the Java programming environment, such as the KVM/CLDC/MIDP software suite, there are severe constraints on code mobility because of the rigidity of class loading. A notable example of this limitation is the impossibility of overriding the embedded class loader to instantiate a newly programmer-defined one.

The main and crucial design choice in the MU middleware is to provide any portable device with one companion entity, called *shadow proxy*, and with several application-specific processors, called *QoS adapters*. Shadow proxies and QoS adapters are implemented as mobile agents, built on top of the SOMA* programming platform. The shadow proxy exploits discovery solutions to retrieve local resources and service components, possibly negotiates tailoring parameters, acts on behalf of the device in case of disconnection, and can follow device independently of their movement between network localities. QoS adapters can operate QoS management operations on VoD flows depending on user/device profiles and can exploit MA mobility to maintain the same physical location of (or at least, close to) the shadow proxies they work for. The components of the MU middleware and their main functions are described in the next section, while the implementation insights in Section 4 focus on how MU shadow proxies can migrate to follow the device change of wireless locality by achieving application-level location visibility in a portable way over different wireless technologies and operating systems.

* Additional information and the code of the Secure and Open Mobile Agents (SOMA) are available for download at `http://lia.deis.unibo.it/Research/SOMA/`

1.1. The MU Middleware Components

The possibility to define suitable and flexible network localities is crucial for the development and deployment of context-dependent Internet services to portable devices. MU offers locality abstractions to describe any kind of interconnected system, from simple Intranet LANs to the Internet (see Figure 1). Any node in the fixed network hosts one place for MA execution; several places can be grouped into domain abstractions, each one usually corresponding to one network locality. For instance, one MU domain typically models one LAN, together with its local IEEE 802.11 service access point and the corresponding wireless LAN cell. In each domain, a default place is in charge of inter-domain routing functionality. MU locality abstractions permit to define loosely coupled localities organized in a hierarchical way; this locality scenario naturally maps into discovery services with the visibility scope of one single domain [5].

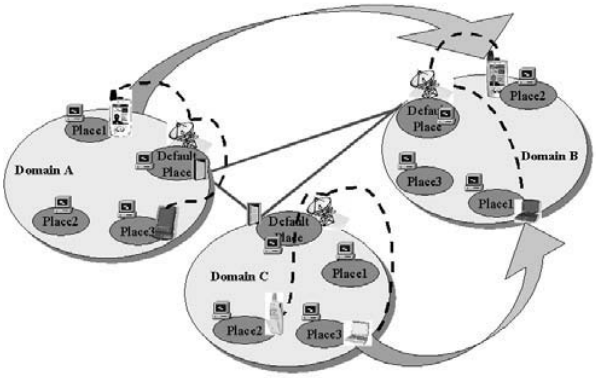

Figure 1. Portable devices roaming among MU network localities.

The MU middleware consists of different components, as depicted in Figure 2.

Shadow proxies represent portable devices over the fixed network and solve the issues related to receiving, caching, and coordinating the tailoring/adaptation of

service results on behalf of the wireless devices with possible intermittent connections and system limitations. Any shadow proxy retrieves the profile of its companion device and its current user. Profiles play a central role in service discovery at the MU Portable Device Lookup Service (PDLS), as described in the following. In addition, the shadow proxy maintains a small amount of VoD flow information in a buffer ahead of playback (as in anti-shock mechanisms for portable CD players), in order to smooth temporary congestion situations in the network connections between the proxy and the server; the buffer size depends on profile information and can change during service provisioning, also in response to the history of session fluctuations in available network resources.

The shadow proxy is implemented as a SOMA agent running within the locality of the MU domain where the portable device is currently attached. During a service session, any shadow proxy automatically migrates to follow its portable device while roaming between different MU domains. It is the MU PDLS that has full visibility of wireless device location (see Section 4) and that triggers the corresponding proxy migration. Several shadow proxies for different devices can execute concurrently on the same place without interfering with each other because the underlying SOMA platform provides isolated execution environments and separated security domains. We usually associate one shadow proxy for each portable device (with a 1-to-1 mapping). However, it is also possible to define group shadow proxies in charge of managing a set of portable devices with synchronization constraints, e.g., when distributing a synchronized VoD flow to a group of tourists walking in the city downtown or visiting the rooms of a museum.

QoS adapters are in charge of compression, e.g., reduction of frame size/rate, and format transcoding, e.g., from AVI to MPEG, on VoD contents to tune the provided QoS level to suit the profile of device characteristics and user preferences. QoS adapters are implemented as SOMA agents that can follow the movements of their shadow proxy. They receive multimedia flows, operate transformations on them and forward processed flows to device-specific VoD players. The current implementation of QoS adapters is based on SUN Java Media Framework (JMF) for multimedia reception, transmission and processing [14]. For the transport and control of packet flows, QoS adapters exploit the JMF APIs to integrate with the Real Time Protocol (RTP) and its corresponding RTCP control protocol [15]. Any MU middleware component is generally portable on any platform that hosts a JVM. For performance sake, QoS adapters sometimes exploit local plug-ins available as native components. To achieve portability, they retrieve dynamically the list of plug-ins installed on their current place of execution to bind only to the available components.

Device/player-specific stubs run on portable devices and are in charge of transferring VoD flow requests from locally installed device-specific players to associated shadow proxies, of receiving multimedia flows and of forwarding them to the local players. We have currently implemented two different stubs. The first one runs on top of the Java standard distribution and interworks with the standard JMF video player. The second one is based on the PersonalJava-compliant Insignia Jeode JVM for HP iPAQ with Windows CE and acts as a stub for the PocketTV MPEG Movie Player [16]. Sometimes, clients may need to have on-line visibility of their location to drive middleware decisions and to take consequent operations of service configuration/management; MU can provide client stubs with full location visibility via the portable solution presented in Section 4.

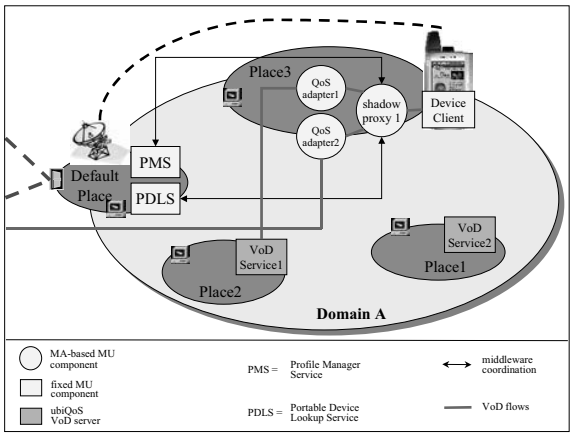

Figure 2. The MU middleware to support VoD distribution to portable devices.

MU includes also two non-moving middleware components, the PDLS and the Profile Manager Service (PMS). They are the only infrastructure components that any MU domain should have installed to participate in the VoD distribution over the wireless Internet.

PDLS is responsible both for sensing when a portable device enters its MU domain (the MU mechanisms to achieve location visibility are extensively discussed in the following section) and for managing tailored lookup requests. Triggered by device arrivals, PDLS asks the SOMA mobility-enabled naming system [6] whether the shadow proxy of that device is already running somewhere in the global system. If the proxy is already active, 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 that can carry the request (the usual lookup services behavior) but only a reference to a suitable QoS adapter able to act as the intermediate between the shadow proxy and the actual service component. PDLS is based on Jini and extends the SUN Reggie reference implementation of the lookup server [17]. It additionally considers the user/device profiles carried by the shadow proxy to identify the needed QoS adapter, then binds it to the requested service component, and finally triggers the adapter migration to the shadow proxy.

PMS maintains profiles of supported devices and registered users. It implements a partitioned and partially replicated directory service specialized for profiles. PMS maintains local copies of profile information and is able to coordinate with PMSs in other domains, via either LDAP or HTTP, to provide global profile visibility to shadow proxies. Device and user profiles are expressed according to the W3C Composite Capabilities/Preference Profile specification [18]. For instance, the MU profile in Figure 3 describes a Windows CE portable device hosting the Insignia Jeode JVM and the PocketTV player. MU permits to express device profiles as deviations from default (vendor-based) ones and to cache distributed copies of diffused profiles, thus reducing significantly the network traffic due to profile exchange during the negotiation phase.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
        syntax-ns#"
    xmlns:ccpp="http://www.w3.org/2002/11/08-ccpp-
schema#"
    xmlns:TermProf="ldap://ldap.deis.unibo.it/TermSch#">
<rdf: Description rdf: about="ldap://ldap.deis.unibo.it/
    MU/MyProf">

<ccpp:component>
   <rdf:Description rdf:about="ldap://ldap.deis.unibo.it/
        MU/TermSW">
   <rdf:type rdf:resource="ldap://ldap.deis.unibo.it/
        Schema#SWPlat">
   <ex:name>WindowsCE</ex:name>
   <ex:version>3.0</ex:version>
   …
</rdf:Description> </ccpp:component>

<ccpp:component>
   <rdf:Description rdf:about="ldap://ldap.deis.unibo.it/
        MU/JVM">
   <rdf:type rdf:resource="ldap://ldap.deis.unibo.it/
        Schema#JVM">
   <ex:name>Insignia Jeode</ex:name>
   <ex:compliance>Personal Java 1.2</ex:compliance>
   …
   <ex:VoDPlayer>PocketTV</ex: VoDPlayer>
   <ex:suppFormat>MPEG</ex: suppFormat>
   …
</rdf:Description> </ccpp:component>
…
```

Figure 3. CC/PP-compliant MU profile for a Windows CE device with Jeode and PocketTV.

The above infrastructure of MU components permits to distribute and tailor VoD flows within one MU domain. This infrastructure is tightly integrated with our active middleware for the support of QoS tailoring, control and adaptation of VoD flows over best-effort networks, called ubiQoS. ubiQoS components can build a dynamic path of active nodes between the VoD flow source and its group of receivers: active nodes negotiate the QoS level on any path segment depending on client requirements, both in terms of user preferences and terminal characteristics. Active nodes monitor and control QoS levels during service provisioning and react promptly with service management operations in case of network congestion. In addition, any active node can cache the traversing VoD flows before their possible local QoS downscaling, thus permitting the realization of an effective distributed caching infrastructure [9]. MU extends significantly the applicability of the ubiQoS middleware to portable access devices with wireless connectivity and strict limits on local resources.

4. Implementation Insights about Portable Location Visibility in MU

The previous section has already sketched that the MU middleware provides location visibility in a portable way over different implementations of wireless technologies over different platforms, to either PDLSs or client stubs. At the state of the art, there is no standard specification, accepted by vendors and spread across the most common operating systems, of the application-level API for cell location visibility in wireless networks. This has produced several vendor-specific and technology-specific solutions, which may significantly limit and slow down the emergence of a wide market of location-dependent services for the wireless Internet. The MU approach achieves portability of the location visibility service by providing different implementation mechanisms, automatically downloaded and deployed depending on the system characteristics of wireless service access points and of client devices.

The PDLS visibility is crucial to trigger shadow proxy migration to follow the client device movements between different wireless-enabled MU domains. The autonomous migration of shadow proxies permits the deployment of MU-based location-dependent services without impacting on the application logic. In other words, migration is completely location-transparent and not visible to the client/server implementation [5].

In the case of Wi-Fi connectivity, MU exploits the monitoring information that IEEE 802.11 service access points make available via standard Simple Network Management Protocol (SNMP) Management Information Bases (MIBs) [19]. In particular, the service access point is configured to notify an SNMP trap to the local PDLS anytime a new portable device associates with the local wireless network locality. This permits PDLS to acquire visibility of all the associated

wireless access devices, and, in particular, to sense and control any new device entering the local MU domain.

In the case of Bluetooth-based service access points (Bluetooth connectivity exploited in the so-called infrastructure mode [20]), PDLS uses the Java Bluetooth API specification to access the information about the devices currently connected to the network locality. The Java Bluetooth API is a novel specification to enable the portable interaction with Bluetooth functions directly from within the JVM [21]. For instance, the `javax.bluetooth.DiscoveryAgent` class provides non-blocking methods for device discovery and the `javax.bluetooth.DiscoveryListener` class permits to define listeners that respond to device discovery events. Let us observe that MU can exploit the Java Bluetooth API also in Bluetooth ad-hoc connectivity modes in a similar way. That would enable a master Bluetooth station to have visibility of its current slave devices; we are currently working at this extension of the MU location visibility service.

It is sometimes necessary to have location visibility also at the client side. For this reason, MU also gives full location visibility to client stubs that can exploit either IEEE 802.11 or Bluetooth connectivity. In more details, for Wi-Fi-enabled client devices with the Linux operating system, MU provides a Java API to use the Linux Wireless Extensions to obtain the list of all service access points currently in the client visibility and some related communication-level information, such as signal strength [22]. In particular, MU accesses this wireless monitoring information via the standardized virtual directory `/proc/net/wireless` provided by the Linux Wireless Extensions. If the client devices host Windows CE 3.0 or Windows CE.NET, MU exploits functions available in the Network Driver Interface Specification User-mode I/O (NDISUIO), which is platform-dependent but portable on any network vendor implementation, to obtain the same information available in Linux [23]. For instance, MU exploits the NDISUIO function `DeviceIOControl()` to query the `OID_802_11_BSSID_LIST_SCAN` object in order to achieve the visibility of the complete list of currently visible Wi-Fi service access points. Finally, in the case of Bluetooth connectivity (as in the case of PDLS) MU exploits the Java Bluetooth API to obtain the list of available Bluetooth base stations. For instance, MU uses methods of the `javax.bluetooth.DiscoveryAgent` class: `retrieveDevices()` obtains the list of all visible Bluetooth devices, used as input to `searchServices()` that returns the set of base stations when queried only for network access point services.

At middleware deployment time, MU exploits the profiles maintained in PMS to choose which ones among the above mechanisms for location visibility have to be installed over PDLS components and access devices, depending on the type of wireless connectivity and on the operating system. In that way, the MU location visibility service achieves portability over a large set of deployment scenarios and permits the installation of the middleware components only where specifically needed, by maintaining the same programming interface for the developers of PDLSs and client stubs. At the PDLS default places, the installation of the location visibility modules is performed by locally migrating specialized SOMA agents; at the client devices, it exploits the code upload mechanisms made available by the Java 2 Micro Edition.

For the sake of testing the usability and effectiveness of the MU middleware in a real application scenario, we have designed and implemented a movie-info service. The prototype supports the accessibility of Wi-Fi devices to information about the movies in the theatres in their current location areas. Any MU domain models a different area and hosts one default place providing IEEE 802.11b connectivity. When a portable device enters a domain, the local PDLS either instantiates or moves there a shadow proxy. In the case of new instantiation, the proxy authenticates the device client and requests device/user profiles to PMS. When the device client requests the movie-info service, the proxy interrogates PDLS by providing user/device profile information. PDLS answers by triggering the migration of a suitable QoS adapter to the proxy. The QoS adapter requests the movie list to a movie-info server available in its domain. The server can maintain locally all the information about the movies currently shown in the area, or can request part of the information, e.g., movie trailers, to distributed remote servers, such as the film producers' Web sites. In the latter case, the movie-info server component local to the QoS adapter acts as the first active node in the active path dynamically established by the ubiQoS middleware components [8]. Let us note that in the movie-info prototype the MU middleware has enabled portable devices to access a context-dependent Internet service without forcing to modify the server design and implementation. In particular, the movie-info servers are standard HTTP servers with hypertexts about the movies currently shown in their area; these hypertexts include multimedia trailers in either WAV format or AVI.

The first experimental results of the movie-info prototype show the feasibility of our approach at the application level. In deployment scenarios with IEEE 802.11 service access points interconnected via 100Mb/s Ethernet, the time needed for our middleware to sense the device change of MU domain and to migrate the associated shadow proxy accordingly varies between 200ms and 400ms, depending on the size (from 10KB to 1MB) of the VoD buffer carried by the

shadow proxy during the migration. This buffer size generally gives enough time to the proxy to re-establish the connection with the VoD server after migration. Therefore, the movie-info user perceives only a VoD flow interruption for the interval due to the proxy migration time. Also this interruption can be avoided/reduced by buffering portions of the VoD stream in advance at the client stub (by exploiting the client-side visibility of the complete list of visible access points and the corresponding signal strength), if portable device memory limitations make it possible.

5. Conclusions and On-going Work

The provisioning of both traditional and context-dependent services over the wireless Internet requires facing several specific support issues, mainly stemming from device mobility, heterogeneity and limited resources. The deployment scenario strongly suggests the organization of a dynamic middleware capable of supporting differentiated QoS levels depending on the specific characteristics of mobile access terminals. This is particularly crucial when dealing with Internet services, such as VoD distribution, that imposes strict requirements on client/network resource availability. MAs are a suitable and effective technology to implement such a middleware because of their intrinsic ability of extending the fixed Internet infrastructure, only where needed, by following the client movements during the service session.

The first encouraging results obtained by the MU implementation, both in feasibility of the application-level approach and in usability of the middleware, are suggesting additional research activities.

On the one hand, we work on portable solutions to estimate wireless device trajectories and to predict the next visited localities. Mobility prediction will enable the MU middleware to move in advance the buffer copies that shadow proxies maintain to smooth temporary congestion. This is crucial for full continuity of VoD streaming, independently of the client roaming, during the time interval needed for the migrated proxies to re-establish the connection to the VoD server. In addition, pushing buffer copies in advance allows to migrate lighter proxies when MU senses the client change of locality. Some first vendor-specific and technology-specific solutions for mobility prediction are at the state of the art of the wireless communication research [24]. To the best of our knowledge, portable implementation solutions for mobility prediction and application-level APIs for their exploitation are still completely lacking.

On the other hand, we start applying the same QoS tailoring approach based on shadow proxies to several different application domains. We work on the development and deployment of context-dependent services that adapt the offered service contents to the client context in general (not only to client location and terminal profile of characteristics, but also to user expertise level/interests/role, computing load and available bandwidth in the visited MU domain). In particular, we are completing the implementation of an Assistant Service for museum visitors that retrieves XML-based information about the artworks exposed in the currently visited museum room.

Acknowledgements

Work supported by the Ministero per l'Istruzione, l'Università, e la Ricerca (MIUR) with the FIRB WEB-MINDS Project "Wide-scale Broadband Middleware for Network Distributed Services" and by the Consiglio Nazionale delle Ricerche (CNR) with the Strategic IS-MANET Project "Middleware Support for Mobile Ad-hoc Networks and their Application".

References

[1] L. Blazevic, et al., "Self Organization in Mobile Ad Hoc Networks: the Approach of Terminodes", *IEEE Communications*, Vol. 39, No. 6, June 2001, pp. 166-174.

[2] C. Perkins (ed.), Special Section on "Autoconfiguration", *IEEE Internet Computing*, Vol. 3, No. 4, July 1999, pp. 42-80.

[3] L. Bos, S. Leroy, "Toward an All-IP-based UMTS System Architecture", *IEEE Network*, Vol. 15, No. 1, Jan.-Feb. 2001, pp. 36-45.

[4] R. Oppliger, "Security at the Internet Layer", *IEEE Computer*, Vol. 31, No. 9, Sep. 1998, pp. 43-47.

[5] P. Bellavista, A. Corradi, C. Stefanelli, "The Ubiquitous Provisioning of Internet Services to Portable Devices", *IEEE Pervasive Computing*, Vol. 1, No. 3, July-Sep. 2002, pp. 81-87.

[6] P. Bellavista, A. Corradi, R. Montanari, C. Stefanelli, "Dynamic Binding in Mobile Applications: a Middleware Approach", *IEEE Internet Computing*, Vol. 7, No. 2, Mar.-Apr. 2003, pp. 34-42.

[7] A. Fuggetta, G. P. Picco, G. Vigna, "Understanding Code Mobility", *IEEE Transactions on Software Engineering*, Vol. 24, No. 5, May 1998, pp. 342-361.

[8] P. Bellavista, A. Corradi, "A Mobile Agent-activated Middleware for Internet Video on Demand", *IPSJ Journal*, Information Processing Society of Japan Press, Vol. 43, No. 11, Nov. 2002, pp. 3301-3315.

[9] P. Bellavista, A. Corradi, "Active Middleware for Internet Video on Demand: the QoS-aware Routing Solution in ubiQoS", *Microprocessors and Microsystems*, Elsevier Science, Vol. 27, No. 2, Mar. 2003, pp. 73-83.

[10] E. Kovacs, K. Rohrle, M. Reich, "Integrating Mobile Agents into the Mobile Middleware", *Mobile Agents Int. Workshop (MA'98)*, Springer-Verlag LNCS, Germany, 1998, pp. 124-35.

[11] S. Lipperts, A. Park, “An Agent-based Middleware: a Solution for Terminal and User Mobility”, *Computer Networks*, Vol. 31, Sep. 1999, pp. 2053-62.
[12] H. Jormakka, et al., “Agent-based TINA Access Session Supporting Retailer Selection in Personal Mobility Context”, *Int. Telecomm. Information Networking Architecture Conf. (TINA'99)*, USA, Apr. 1999, pp. 68-76.
[13] IKV++ Technologies AG - *enago mobile*, http://www.ikv.de
[14] Sun Microsystems Inc. - *Java Media Framework (JMF) API*, http://java.sun.com/products/java-media/jmf/
[15] R. Braun, “Internet Protocols for Multimedia Communications II: Resource Reservation, Transport, and Application Protocols”, *IEEE Multimedia*, Vol. 4, No. 4, Oct.-Dec. 1997, pp. 74 –82.
[16] MPEG TV LLC – *Pocket TV*, http://www.mpegtv.com/wince/pockettv
[17] Sun Microsystems Inc. - *Jini*, http://developer.jini.org
[18] WWW Consortium - *Composite Capabilities/Preference Profile (CC/PP)*, http://www.w3.org/Mobile/
[19] M. Gast, *802.11 Wireless Networks: the Definitive Guide*, O'Reilly, Apr. 2002.
[20] P. Johansson, et al., "Bluetooth: an Enabler for Personal Area Networking", *IEEE Network*, Vol. 15, No. 5, Sep.-Oct. 2001, pp. 28-37.
[21] The Java Community Process - *Java APIs for Bluetooth (JSR-82)*, http://www.jcp.org/en/jsr/detail?id=82
[22] Debian - *Tools for Manipulating Linux Wireless Extensions,* http://packages.debian.org/stable/net/wireless-tools.html
[23] Microsoft Software Developer Network - *NDIS Features in WindowsCE*, http://msdn.microsoft.com/library
[24] Yu Fei, V.C.M. Leung, “Mobility-based Predictive Call Admission Control and Bandwidth Reservation in Wireless Cellular Networks”, *IEEE INFOCOM 2001*, pp. 518-526.