Information Systems Frontiers 6:1, 23–34, 2004
© 2004 Kluwer Academic Publishers. Manufactured in The Netherlands.

The Mobile Application Server (MAS): An Infrastructure Platform for Mobile Wireless Services

Ravi Jain*
DoCoMo USA Labs, 181 Metro Drive, Suite 300, San Jose, CA 95110, USA
E-mail: jain@docomolabs-usa.com

Stefano Puglia
Nexse Software Engineering, Circonvallazione Appia 113, 00179 Rome, Italy
E-mail: stefano.puglia@nexse.com

John Wullert, Kirthika Parmeswaran and John-Luc Bakker
Telcordia Technologies, Applied Research, 445 South Street, Morristown, NJ 07960, USA
E-mail: jwullert@telcordia.com
E-mail: kparmesw@telcordia.com
E-mail: jbakker@telcordia.com

UNCORRECTED PROOF

Abstract. *There has been a surge of interest in developing advanced mobile voice and data services. One of the biggest challenges faced by service providers is to create new services efficiently and rapidly. We describe the motivation, architecture and design of the Mobile Application Server (MAS), a Java middleware prototype platform for mobile services which provides basic common building blocks required by mobile wireless applications, such as user profile management, location handling, device detection, content adaptation and e-wallet functions. The MAS is modular, flexible, and provides an open API for 3rd party service creation. Users can thus utilize a variety of access technologies (CDPD, cellular, 802.11, SMS, etc.) and delivery mechanisms (HTML, WML, VoiceXML, etc.) while application developers are spared the details of this heterogeneity. The MAS also interfaces to traditional telecommunications system (A/IN) elements such as SCPs as well as Next Generation Network (NGN) or converged-network elements such as Call Agents. We present application examples of the use of MAS, initial performance measurements and discuss some lessons learned from this experimental system. We conclude with directions for further work.*

Key Words. *mobile middleware, wireless services, pervasive computing*

1. Introduction

A core technological glue of virtual enterprise infrastructure is the ability of users to communicate anytime, anywhere, via any device. The users in such enterprises of the future will have access to a wide range of mobile wireless technologies for their communication, e.g. ranging from PCS or cellular systems to wireless LAN and Bluetooth systems, even if some technologies fail, are subsumed by others or find only niche markets. Faced with this heterogeneity in wireless media and end devices (pagers, cell phones, PDAs, laptops, etc.), users will desire services that operate seamlessly across multiple technologies and provide advanced value-added features at acceptable cost.

So far, the explosion in the mobile wireless PCS/cellular market has largely been driven by simple wireless voice and data services, such as two-party voice calls and short messaging. However, it is clear that the per-unit revenues (e.g. dollars per minute) from these simple services are declining and are expected to continue doing so (Cellular Telephony Industry Association, 2001). Service providers have made large investments in spectrum and basic infrastructure for 3G wireless systems in anticipation of providing advanced value-added services rather than simply serving as

*To whom correspondence should be addressed.

conduits of bits. Similarly, market research indicates that the availability of additional applications is a key motivator for mobile data users (Telephia Inc., 2001). However, typically value-added services have been developed in an evolutionary manner adding new features to these basic services. Alternatively they have emerged by developing specialized vertical applications (e.g. mobile sales or field service) for particular markets. Similarly, wireless LANs have either been used for basic wireless data access (e.g. to the Internet or to legacy enterprise applications), or for specialized applications developed for niche markets (e.g. in hotspots like airports or factories).

Thus a critical challenge for service providers is to offer advanced value-added services rapidly, efficiently and seamlessly across heterogeneous wireless technologies. This is compounded by the fact that despite the proliferation of new mobile wireless services being introduced in the marketplace, it is not yet clear which will be the "killer applications", for which market segments and in which regions. Further, it is also clear that at least for the medium term mobile wireless services will continue to utilize the information and capabilities of existing telecommunications systems and network elements, such as Service Control Points (SCPs) in the (Advanced) Intelligent Network, as well as Next Generation Network (NGN) elements like softswitches and Call Agents. (For background on NGN architectures see, for example, Huitema et al. (1999)).

This challenge calls for a service infrastructure that is *modular*, allowing software components to be reused for multiple services; *flexible*, allowing new software components to be introduced easily for new classes of services as well as new wireless media and devices; *evolutionary*, interfacing to A/IN as well as NGN elements; *content-independent*, in the sense of not being tied to or providing content itself but easily leveraging third party content; and *open*, allowing third-part application developers to develop services rather than only infrastructure vendors or the service providers themselves.

Examples of common mobile wireless applications that are being developed and marketed include location aware services (e.g. finding directions to the destination depending on the current location), information services (e.g. stocks, sports and weather information), and mobile shopping. These services need both to be deployed on demand by the service providers and to be adapted or "morphed", ideally according to the end device's displaying capabilities, bandwidth, location, movement and other factors.

Our experience with developing mobile wireless applications and platforms suggests that:

- there is a core set of "horizontal" features that many users desire, such as unified messaging, personalized information delivery (news, weather, etc.), wireless voice, wireless web access, etc.;
- most features for horizontal as well as vertical applications, in turn, rely on a common set of building blocks such as user profile management, location and end device information management, authentication, and electronic wallet facilities;
- providing these facilities over wireless links to users who may be mobile and use different devices with different media at different locations requires adaptation and customization for heterogeneous devices and media.

In this paper our contribution is to describe an effort to build on these observations and develop an architecture and system for a mobile wireless services platform that meets the five requirements described above: being modular, flexible, evolutionary content-independent and open. While these requirements seem rather generic, translating them into a pure Java working experimental prototype using only currently commercially available software technology (Enterprise Java Beans (EJB) (Monson-Haefel, 1999), XML (W3C, 2000, http://www.w3.org/TR/REC-xml), XSLT (W3C, 1999, http://www.w3.org/TR/xslt), WML (The WAP Forum, http://www.wapforum.org), VoiceXML (W3C, 2000, http://www.w3.org/TR/voicexml/), XHTML (W3C, 2000, http://www.w3.org/TR/xhtml1/), etc.) is non-trivial. In addition, while there have recently been numerous commercial announcements of infrastructure platforms for mobile wireless services, it is not clear which of them demonstrably meet all five requirements, since details are not always made public. Also, while numerous mobile middleware research platforms have been built, to our knowledge there is little published in the research literature that describes a system that shares our goals, approach, design and implementation (e.g. see Schilit et al. (2001), Shklar et al. (2001), Cocoon, (http://xml.apache.org/cocoon/index.html) AT&T (to appear) and Joshi (2000) and references therein). Finally, we provide a brief summary of our experimental

UNCORRECTED PROOF

experience including strengths and limitations of the technology used, impact of emerging technologies such as Web Services, and open issues that remain to be addressed.

The rest of this paper is organized as follows. A quick overview of commercial mobile middleware platforms is provided in Section 2 for background information. Section 3 describes the system architecture of the Mobile Application Server (MAS) experimental research prototype, including its capabilities and key interfaces. Section 3 describes how the MAS is used to build example services, including a location-based service that uses VoiceXML technology. Section 4 provides a discussion of our experimental insights and Section 5 of future work and emerging technologies. We end with brief concluding remarks.

2. Overview of Mobile Application Middleware Platforms[1]

Simply stated, *Mobile Application Middleware Platforms* (*MMAP*) package several technologies (wireless network support, wireless middleware services, wireless software development environments, and wireless monitoring/control systems) to build, deploy and manage mobile applications. Nokia's WAP Server, Oracle's Mobile Application Server, and Agilo's Server IBM's Web are examples of MMAP (see the sidebar "Examples of Commercially Available Mobile Applications Middleware Platforms"). Figure 1 shows the conceptual components of a MMAP. A MMAP is part of a multi-tier (mostly three tier) architecture that typically consists of a thin client tier residingon handheld devices, a middle tier consisting of business applications and a set of middleware and network services, and a backend tier consisting of mission-critical databases and applications. The key components of a MMAP are:

- *Mobile Application*. The purpose of a MMAP is to support a mobile application. This application contains the business functions specifically developed for the mobile users and typically needs integration with backend database or business application systems such as mainframe financial accounting systems, manufacturing systems, inventory, ERP (Enterprise Resource Planning) and CRM (Customer Resource Management) systems. This application may be a MEBA (Mobile Ebusiness Application) or M-P-T-V (Mobile, Positional, Television, Voice) Commerce.
- *General Purpose (Back-end) Middleware*. To support the back-end systems, common back-end middleware services such as database and transaction processing are needed.
- *Mobile Devices*. This may be a cellular phone, notebook, hand-held computer, pen computer, PDA, PalmOS compatible PDA, Windows CE/Pocket PC device, or a two-way interactive pager. These devices operate under different mobile operating systems that reside in the mobile device; it may be Windows98/2000/NT, PalmOS, Win CE (or Pocket

UNCORRECTED PROOF

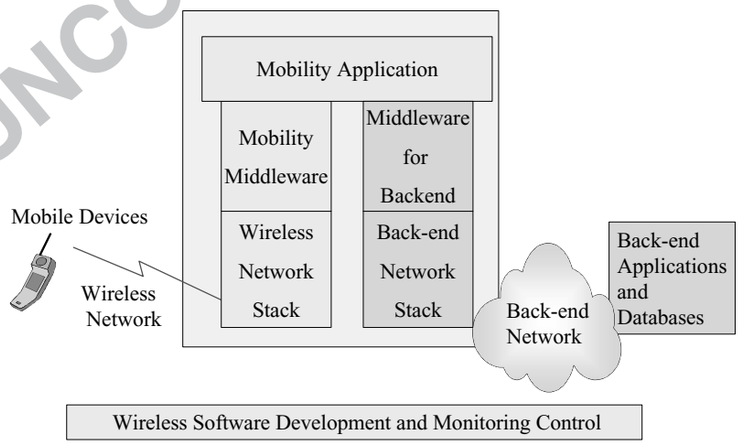

Fig. 1. *Mobile Application Middleware Platform (MMAP)—Conceptual view.*

PC), EPOC, a specialized OS like Blackberry, or a voice/web browser.

- *Wireless Network.* This may be a wireless LAN, WAN or MAN. These networks are provided by companies such as Cingular (formerly Bell South Wireless Data), Verizon, Sprint, Metricom, Nextel, Bell Mobility (Canada), AT&T (Canada), BT in UK, and NTT DoCoMo (Japan). To connect computing devices to the wireless network, you need a suitably-configured wireless WAN modem, wireless LAN adapter, or wireless MAN (metro-area network) adapter. While the wireless network provides true mobility, you may utilize a wireline network for those mobile users who need occasional wired connection from hotels, motels or airport first-class lounges. Some of these airports are now offering wireless LAN connectivity to wireline backend networks.
- *Mobility Middleware*. The mobility middleware is responsible for handling mobile devices and is the heart of a MMAP. This middleware takes raw data from database applications/queries and transforms the data to a specific thin client (or a thick client like a PC) considering its presentation space characteristics and limitations. It includes breaking the messages into smaller chunks, filtering redundant information, and even logically compressing the data, etc. This middleware also provides mobility specific services such as location and supports application programming level interfaces (APIs) with specialized communications protocols for wireless.
- *Wireless Software Development and Monitoring/control Facilities.* Special facilities, such as Software Development Kits (SDKs) are needed to build the mobile applications of the future. These SDKs provide GUI development tools to build mobile application modules that use the wireless middleware APIs. In addition, mobile applications need to be monitored and controlled at run time for errors and load balancing. Wireless software development and monitoring/control facilities are very closely tied to the wireless middleware and are usually packaged with them. For example, the Nokia WAP Server has an SDK for WAP and monitoring control commands for WAP.

Within the conceptual model presented above, a wide range of MMAPs from different vendors are available commercially as shown in the sidebar. Some mobile application servers are generic web servers with an SDK (Systems Development Kit) or API capability to pull data from back-end systems and send it to a browser-based client software in a handheld device. Other application servers provide a business application with customization capability for wireless access. Still other mobile application servers are based on mobility middleware such as WAP, I-Mode, or Wireless Java. Depending on the heritage of the vendor and their core expertise, you can categorize application servers in the following broad classes:

- Generic application servers with a web-based SDK with support for handheld devices and wireless networks. Examples are the Netscape, Microsoft, Sun, and BEA application servers with support for wireless devices.
- Generic database servers with support for mobile devices. Oracle MAS is an example.
- eBusiness/eCommerce Application Servers that have been built specifically to support EC/EB applications. Mobility has been added as a feature to these severs. IBM's WebSphere is an example.
- Mobility specific application servers that were built primarily for mobile applications. Nokia's WAP Server or MobileAware Everix are two examples.

Examples of commercially available Mobile Applications Middleware Platforms (MMAP)

- Aligo M-1 Mobile Application Server (www.aligo.com) uses the wireless Java environment, specifically Sun J2ME (Java 2 Micro Edition), to provide the resources and services that developers and architects need to architect, build and deploy Java applications for mobile devices.
- IBM's WebSphere Application Server (www.ibm.com) has the facilities for large-scale mobile EB application development for large enterprises.
- Jacada Presentation Server for Palm (www.jacada.com) addresses the needs of accessing mission-critical information from enterprise systems to the web through handheld devices.
- Nokia's WAP Application Gateway (www.nokia.com) uses WAP and has an extensive array of facilities for mobile device support with connections to back-end applications.
- Oracle's Mobile Application Server (www.oracle.com) allows access to Oracle databases and application suite from mobile devices.

UNCORRECTED PROOF

- Sybase Mobile Application Studio (www.sybase.com) provides a studio of tools for wireless access to Sybase and other databases.
- @Hand Mobile Application Server (www.@hand.com) provides core infrastructure, administrative services and a rapid application development environment for creating mobile applications.
- Volantis Mariner (www.volantis.com) is an Application Server-based solution that enables organizations to deploy scalable web applications to be accessed from a multitude of Internet capable devices. Mariner is built upon application servers such as BEA WebLogic, ATG Dynamo, and IBM WebSphere.
- MobileAware Everix (www.mobileaware.com) enables companies to deliver content in a multichannel environment and is developed to provide full support to open enterprise standards such as J2EE and XHTML.

3. System Architecture

We first describe the place of the MAS within the larger telecommunications and Internet space (see Fig. 2). The MAS has three sets of interfaces: to application developers, to end users, and to third-party content or functionality providers; we describe these below.

MAS provides a set of building blocks, or components, encapsulating functions that are commonly required by mobile wireless services, such as user profile management, message management, location information management, etc. These functions are encapsulated as functional Managers; we will describe them briefly later. Each of these functions exposes one or more pure Java Application Programming Interfaces (APIs), which collectively form the MAS API. Care had been taken to abstract the APIs at a level that encourages rapid application development through maximizing reuse of building blocks and minimizing knowledge of the capabilities that are controlled by the building blocks. Basically non-telecom experts must be able to develop applications or aggregate capabilities using MAS. This API is open in the sense that it is fully documented and uniformly available to any application developer using the MAS as a service execution environment. Application development (traditionally called service creation in telecommunications parlance) can be done in Java (using Java Beans or EJB) or a high-level XML-based scripting language e.g. SCML (Bakker and Jain, 2002).

The second set of interfaces is towards the end user, allowing users to access mobile wireless services executing on the MAS via a variety of wireless media and devices. Supported wireless media include

UNCORRECTED PROOF

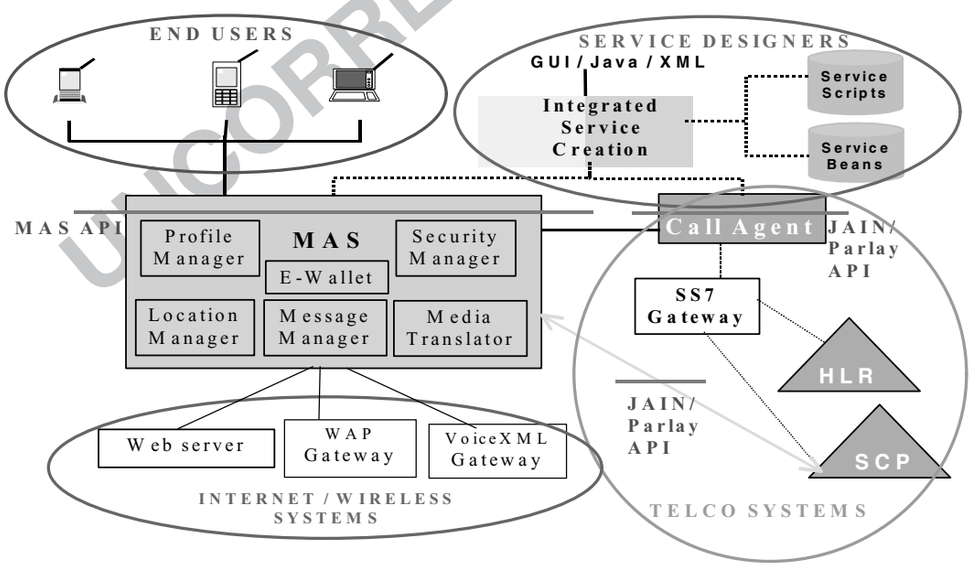

Fig. 2. *The MAS architecture and network environment.*

paging, SMS, CDPD, cellular/PCS, and 802.11 wireless LAN; support for Bluetooth is in progress. Supported mobile devices include one-way and two-way pagers, PDAs, cell phones, and laptops. Supported presentation mechanisms include voice calls, text and voice messaging and chat, video, HTML, XML, WML via WAP, and VoiceXML. In addition traditionally wired access (via fax, email, desktop PC) is also supported.

Finally, MAS interfaces to third-party content providers as well as functionality providers. Examples of content providers are information services (stocks, weather, entertainment), yellow pages, and location and map services. Examples of functionality providers are A/IN network elements such as the SCP or HLR, that can manage location information or perform phone number translations; NGN elements such as softswitches or Call Agents that can provide voice call setup and session handling; and Web-based servers that can provide WAP or VoiceXML gateways or provide functions such as micropayment services. Currently MAS has interfaces to several content providers, such as Mapblast (www.mapblast.com), as well as some web-based functionality providers, such as BeVocal (www.bevocal.com), via standard HTTP connections. Interfaces to A/IN and NGN elements are more complex. The MAS currently interfaces to an SCP (the Telcordia ISCP product) via a proprietary interface, although in future we expect to migrate this to its standard JAIN/3GPP OSA/Parlay or Parlay X interface (Sun Microsystems, 2001, The Parlay Group (http://www.parlay.org), The 3GPP OSA (http://www.3gpp.org), Parlay X (http://www.argreenhouse.com/parlayx/). It also interfaces to an NGN element (the Telcordia Mini Call Agent experimental research prototype) via a JAIN/3GPP OSA/Parlay interface.

A three-tier architecture has been used to implement MAS (see Fig. 3). The architecture consists of a thin presentation layer consisting of servlets and Web pages to obtain user input; a middle-tier containing the business logic, which is a framework of EJBs; and a back-end database for persistent storage. The reasons for selecting this well-known and documented architecture include extensibility, flexibility and programmability.

We briefly describe the functions provided by the MAS components. The middle tier consists of various Managers, implemented as EJB, that encapsulate the business logic to access the persistent structures (user profile, location, etc.) and, effectively, shield the applications from manipulating the structures directly. The Provider is the initial component that applications must go through to access the other MAS components. The Application Manager provides applications with support for the generic lifecycle of a service request for all services (login, activation, etc.) The Profile Manager stores and allows applications to manipulate user profiles including, for example, an addressProfile, contactProfile, and mediaProfile indicating service subscription information for the user as well as the devices and media she may use at different times. The Location Manager stores and provides application access to user location information in symbolic or coordinate (i.e. lat/long) form and supports a variety of location-dependent queries (find the user, find all users in a given area, etc.). To that extend, the location manager can interact with an HLR or directly with the end point to obtain e.g. GPS provided coordinates. The Media Manager performs user device capability detection and adaptation of content. Note that content is available in many different formats, e.g. web content is available in HTML and typically assumes 1024 by 768 pixels. Mobile devices such as PDAs and handsets have a much smaller presentation screen. Based on the information contained in the user's profile(s), MAS converts content such as HTML (or any XML) using XSLT to WML or even VoiceXML. Internally, MAS adapts the content such that it can be represented on the end device, taking into account its particularities. Finally the Security Manager and E-Wallet components provide basic authentication and electronic cash transaction capabilities, respectively.

4. Example Applications

4.1. Voice direction query

In this section we present an example of how an application can use the MAS functions. The following example (Fig. 4) shows a user requesting directions to nearest coffee shop.

A user interaction device translated the voice request into a HTTP POST request (step 1). A Java servlet processes the request and initiates the service (steps 2, 3). To locate the nearest coffee shop and calculate the directions, the application needs to obtain the location of the requesting end user. It obtains a reference to the Location Manager and queries the location (steps 4, 5). The location manager continually or on-demand

UNCORRECTED PROOF

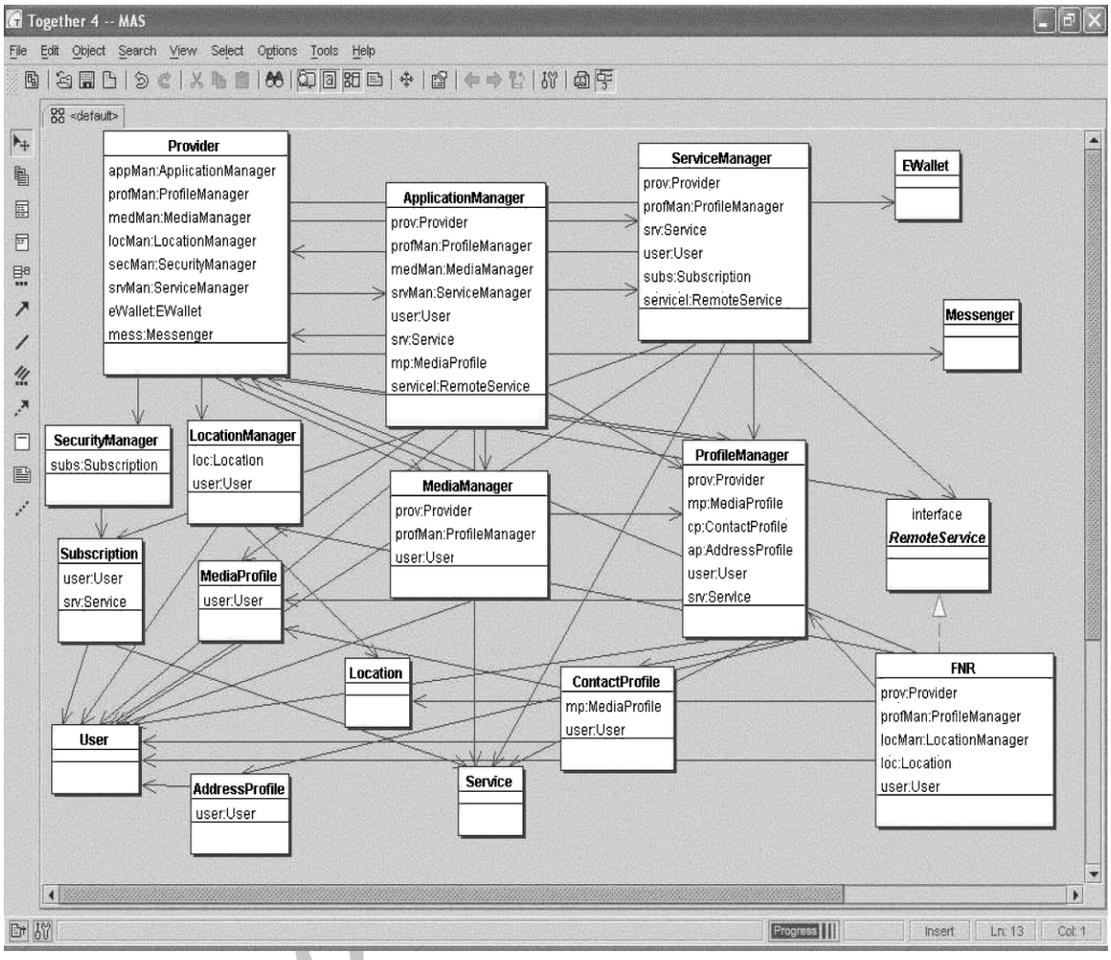

Fig. 3. *Internal architecture of the MAS.*

obtains the actual location through interaction with GPS daemon that is collocated with the end point, but the application need not be aware of the implementation details. Subsequently the application gets content from a third-party content provider (steps 6, 7) and converts it to XML. The Media Manager (steps 9, 10) adapts the content to the user's device (a voice browser) using XSL style sheets, and the resulting VoiceXML is delivered via a voice gateway to the user. We have used third-party web-based voice gateways but due to firewall restrictions are currently using a Nuance gateway product.

In Fig. 5 we provide a UML sequence diagram that shows some more technical detail to illustrate the internal functioning of MAS. A typical user session consists of 3 steps: the user logs in to the service provider (in this case handled by the MAS), is presented with a list of available services, and then invokes a service (e.g. a voice direction query). We show the component interactions for the second step only for brevity, and focus on the server side only. After the user has been authenticated, the MAS servlet (*MasServlet*) issues an API call to the Application Manager (*ApplicationManagerBean*) to retrieve the list of user services. The Application Manager, in turn, requests an instance of the Service Manager (*ServiceManager*) to the Provider (*Provider*) and then invokes the specific method

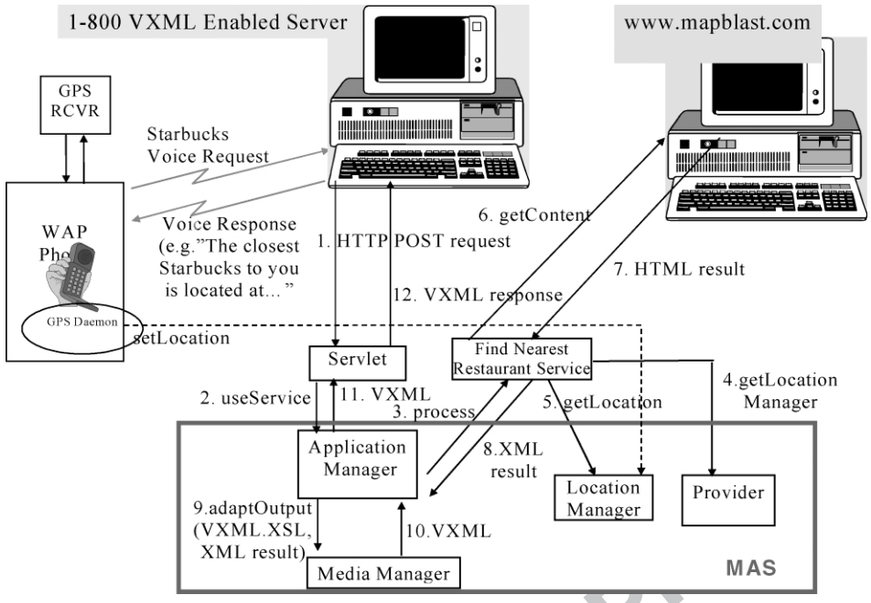

Fig. 4. *Call flow for a voice direction query application.*

(*getSubscribedServices*) to get all the services the user has subscribed to. Once the list is available in a neutral XML string, the Application Manager requests the Media Manager (*MediaManager*) to prepare the final format suitable for the user's device based on the user's media profile (*MediaProfile*) information and on the appropriate selected stylesheet (method *formatOutput*). In this fashion final device-dependent markup languages (i.e. HTML, WML, VoiceXML) can be generated.

4.2. More complex applications

The voice directions query application described in the previous section can and has been extended in various ways, allowing significant reuse of the MAS building blocks. We have developed an application that provides directions by WAP, and in addition provides a hyperlink on the user phone display to allow placing a call to the restaurant (e.g. for reservations) either directly via the service provider's wireless infrastructure, or by the MAS interacting with a Call Agent and performing a third-party call setup using the JAIN/Parlay call control API on top of the Telcordia mini Call Agent prototype. Similarly we have developed an application allowing a user to purchase data services using her wireless voice prepaid account maintained by the SCP (a Telcordia SCP product in our testbed.) Finally, we have developed and demonstrated more sophisticated services, including mobile wireless access to the Telcordia Next Generation Contact Center (NGCC) offering, as well as a movie-by-location application where video clips of movies playing in the closest cinemas (or images for users with constrained bandwidth) are downloaded and users can purchase tickets.

5. Performance Measurements

We have conducted preliminary performance experiments on our implementation of the MAS in order to obtain initial estimates of performance. For brevity we present results for the step where the user requests the list of subscribed services and is presented with such a list based on her output device (i.e., HTML, WML, VoiceXML).

In our experimental setup the MAS is run on a desktop PC which is a Pentium 550 MHz machine with 384 KB RAM operating in our lab network environment. A client generates a workload of requests that is presented to the MAS; to represent a

UNCORRECTED P

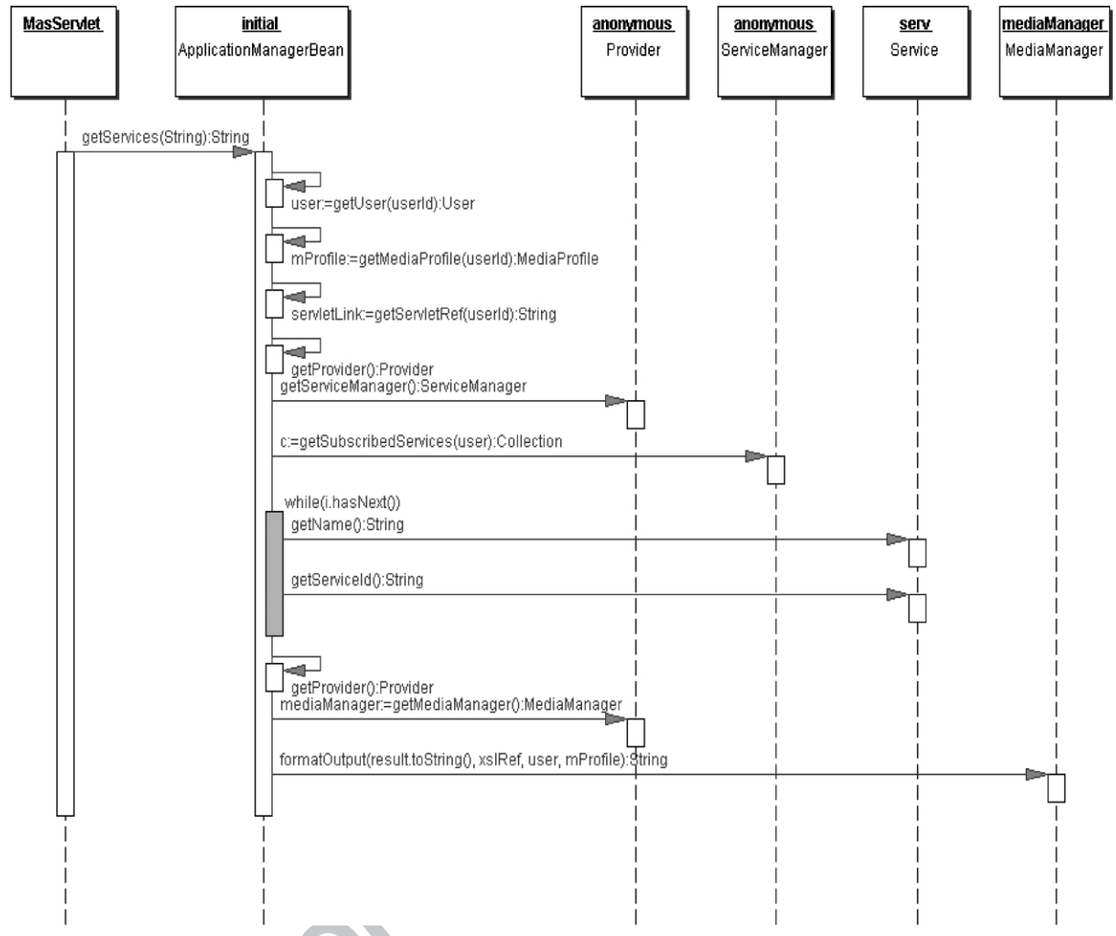

Fig. 5. *UML diagram showing MAS internals for obtaining the list of user subscribed services.*

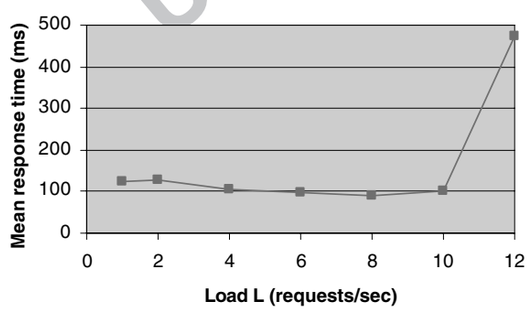

Fig. 6. *Mean responce time for MAS.*

realistic scenario each request randomly requires one of the three possible output formats (HTML, WML, VoiceXML). The workloads we experimented with ranged from $L = 1$ request/sec to $L = 12$, and the metric used was the total response time, i.e., the delay between issuing the request at the client and receiving the corresponding response. Five runs were carried out for each value of workload and the mean response time R (ms) was calculated. Figure 6 shows that for this class of machine we obtain an observed mean response time of about 100 ms up to a workload of $L = 10$. It is clear that for this class of machine in our setup a workload of about 10 requests/sec is the limit in terms of acceptable user response time. We point out that the MAS platform has not been optimized for performance and

the measurements presented here are to indicate trends in the performance.

6. Discussion and Lessons Learned

6.1. General

We find that the interfaces to content and functionality providers are typically proprietary (non-standardized, diverse, vendor-specific), and not always well-specified. In contrast, while the wireless media and devices utilized by end users are also of many types and quite diverse, standardization and (relatively) rigorous and open specification make interfacing to them much easier than to content and functionality provider.

However, we do see a consolidation in progress and expect that web content will move away from HTML-only and will be made available in more standard and semantically-rich forms like the Web Services Description Language (WSDL) (W3C, 2001) for ease of automatic processing. Currently, MAS it is at the mercy of format and stylistic changes made by the content provider. In a true field deployment the service provider would have to enter into appropriate business arrangements with the content provider to obtain advance information about such changes. However, these would be required for purposes other than adaptation, but also billing etc. Additionally, most functions to be performed by present and future telecommunications networks will be accessible through open, published APIs such as JAIN/Parlay, 3GPP OSA, or Parlay X.

The MAS makes full use of a wide variety of standard, maturing technologies both in terms of the general approach that was adopted to design and implement the architecture's functionalities and in terms of languages used to exchange, transform and represent content coming from third parties and delivered to various users' end devices. In particular we demonstrated and verified how a fully J2EE compliant architecture, running on a standard-compliant application server (JBoss 3.0.0, integrated with Jetty as servlet server) and interacting with an all Java, object-relational database management system on the backend can successfully work using the most relevant markup languages (XML, XSL, XSLT, XHTML, HTML, WML, VoiceXML) proposed by the W3C. It is noteworthy that all the software components mentioned above were implemented using publicly available royalty-free software; thus software licensing fees themselves may not be a significant barrier for entering this market.

With respect to the application examples described in the previous section (and others omitted for lack of space), once all the internal machinery to perform the device detection, content adaptation and location handling was implemented in the shape of standard EJBs with remote interfaces, it was straightforward to develop new services (location-dependent or independent), and package them in specific EJBs to be deployed on the platform, as well as to add new MAS facilities as components. For example, a Messenger application allowing cell phone users to send fax and email from WAP phones was rapidly developed utilizing existing MAS components. Also, the integration of the Telcordia Orator text to speech functions proved to be straightforward.

The modular and flexible design of the MAS was useful when upgrading its facilities from delivering only HTML and WML to include VoiceXML. Once a standard markup language for specifying voice dialogs was available, the possibility of supporting service interaction via voice was immediately guaranteed by simply adding stylesheets, built on templates used for other presentations (i.e. WML and HTML), to generate VoiceXML specific tags which can be parsed and executed by any VoiceXML compliant voice web browser.

6.2. VoiceXML

There are differences between VoiceXML and visual mark-up languages that are fundamentally related to the spoken medium. On the output side, while speech synthesis is constantly improving, it is still undesirable to have all prompts delivered in synthetic speech. Thus, while the prompting text can be constant between various text-based systems, the VoiceXML implementation requires the use of pre-recorded audio prompts.

There are differences on the input side as well. In particular, there is really nothing within an HTML page or WML card that corresponds to the grammar required for speech input. While a pull-down menu of options might represent a list of choices, such a list would not represent all the possible inputs and would make a poor grammar even for the options that were represented. Thus the VoiceXML style sheets need to contain input and output items that are not necessary for other mark up languages. Further, while menu-driven interfaces with a few choices are well suited for the small screens on HTML browsers or WAP phones, and in

UNCORRECTED PROOF

principle can work for voice interfaces, for some applications voice can be made much more powerful by using natural language mode.

Our approach allows menu-driven and natural language interfaces to co-exist. We initially constructed a menu-driven VoiceXML interface that requires users to step through a tree to get to the functions they desire. We then created a natural language grammar for the main VoiceXML page that allows users to bypass the branches of the tree and jump immediately to the desired functions by mapping the various natural-language phrases to URLs that correspond to the various functions. We are currently experimenting with both menu-driven and more natural language interfaces.

6.3. Use of XSL

The use of XSL stylesheets for content adaptation was found to have benefits as well as limits. XSL is relatively easy to read and use for non-Java programmers, and is thus commonly being used for content presentation on the Web. However, Java Server Pages (JSP) are advocated as an alternative by many J2EE 'purists'. JSP is easier to use for Java-skilled programmers, and reasonably comparable to XSL in terms of the amount of code, but binds all the transformation machinery to a specific language. XSL itself amounts to a programming language, and one that does not have a very readable structure; we also discovered incompatibilities among the currently available XSLT processors. XSL documentation tools are becoming available e.g. xsldoc (http://www.xsdoc.org) that produce programmer-friendly documentation in a manner similar to Javadoc (http://java.sun.com/j2se/javadoc), but the documentation tags are not standardized. Finally, both JSP and XSL can be compiled such that they don't have to be parsed and executed for each transformation request. XSL stylesheets are fairly efficient when compiled into Java bytecode (Apache makes an XSLT compiler (http://xml.apache.org/xalan-j/) available).

7. Future Work and Conclusions

Recently, there has been much interest in web services as well as the W3C Composite Capabilities/Preferences Profiles (CC/PP) (www.ccpp.org) and the Resource Description Language (RDF) (http://www.w3.org/RDF). Here we discuss these developments briefly and their impact on future work on the MAS platform.

Web services can be defined in WSDL and advertised using UDDI (Universal Description, Discovery & Integration) (http://www.uddi.org/) or WSIL (Web Services Inspection Language) (http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html). WSDL is typically employed by systems that provide raw access to data without any visual markup. As MAS is a portal positioned at the intersection between the Internet, mobile and switched networks; it is perfectly suited to deal with WSDL and generate target HTML based on the constraints caused by the end user's device and access network. The additional advantage is that WSDL interfaces are designed to deal with backwards compatibility and are less prone to periodic visual redesign. Additionally, WSIL, in combination with e.g. HTTPS, makes authentication and authorized access to web services possible.

CC/PP in combination with RDF (Shklar, 2001) is a way to specify what exactly a user agent (web browser) is capable of presenting. Making MAS CC/PP and RDF aware will position it as a proxy for markup transformation, transmission, or caching services for CC/PP-enabled clients and servers. Currently, MAS selects the appropriate transformations based on information inferred from the initial request, e.g. from the http request header.

We conclude that service providers face a challenge in delivering advanced value-added mobile wireless applications rapidly and efficiently. We have described the internal architecture of a experimental prototype middleware platform called the Mobile Application Server (MAS) that utilizes current software technology to address this challenge. We have also described its external interfaces and connections to A/IN as well as NGN elements, and demonstrated its use via simple example applications that have been implemented. The latter half of the paper focuses on the lessons learned from this experiment and the strengths and limitations of some of the technology used, in particular VoiceXML and XSL, as well as directions for future work.

We have shown that the MAS' service infrastructure is *modular* and *flexible*; composed of a set of abstract services implemented as EJBs. EJBs for new or missing functions can easily be added as has been demonstrated in the course of the project. The *evolutionary* aspects of MAS were demonstrated as it interfaces both with

UNCORRECTED PROOF

in-house call agent as with a commercial grade A/IN network element. One of the main functions is the *content-independent* aspect of MAS; it leverages readily available, (third party) content to a host of end user devices. Finally, the MAS APIs are *open* in the sense that they are well documented and available to third parties for developing and deploying their applications.

Note

1. This discussion is largely based on Umar, A., "e-Business and Distributed Systems Handbook: Platforms Module," January 2003, ISBN: 0-9727414-7-X, available through amazon.com or www.amjadumar.com.

References

AT&T, iMobile EE—An Enterprise Mobile Service Platform. *ACM/Kluwer Journal on Wireless Networks* (to appear). See: http://www.research.att.com/sw/tools/imobile/.

Bakker J-L, Jain R. Next generation service creation using XML scripting languages. *Proc. IEEE Intl. Conf. Comm. (ICC)*, to appear, June 2002.

CC/PP. See www.ccpp.org.

Cellular Telephony Industry Association, Semi-Annual Wireless Industry Survey, 2001. See http://www.ctia.org/.

Cocoon, see http://xml.apache.org/cocoon/index.html.

Huitema C, Cameron J, Mouchtaris P, Smyk D. An architecture for residential internet telephony service. *IEEE Internet Computing*. May–June 1999;3(3).

Joshi A. On Proxy Agents, Mobility, and Web Access. *ACM/Baltzer Journal of Mobile Networks and Nomadic Applications (MONET)*, Dec., 2000.

Monson-Haefel R. *Enterprise JavaBeans*. O'Reilly, 1999.

Parlay X: http://www.argreenhouse.com/parlayx/.

Schilit BN, Trevor J, Hilbert D, Koh TK. m-Links: An infrastructure for very small internet devices. In: *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, 2001: pp. 122–131.

Shklar L. RDF models for dynamic syndication and wireless applications. *Semantic Web Workshop*, May 2001.

Sun Microsystems. The JAIN APIs, August 2001. See http://java.sun.com/products/jain/.

Telephia Inc., *New Applications Key for Mobile Devices, m-business*, March 2001: p. 36.

The 3rd Generation Partnership Project (3GPP). Open Services Architecture (OSA). See http://www.3gpp.org.

The JavaDoc Tool Homepage. See http://java.sun.com/j2se/javadoc.

The Parlay Group. See http://www.parlay.org.

The WAP Forum. See http://www.wapforum.org.

UDDI, see http://www.uddi.org/.

Xalan-Java. See http://xml.apache.org/xalan-j/.

XSLDOC, see http://www.xsdoc.org.

W3C, XSL Transformations (XSLT) Version 1.0. W3C Recommendation, November 1999. See http://www.w3.org/TR/xslt.

W3C, Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation, October 2000. See http://www.w3.org/TR/REC-xml.

W3C, Voice eXtensible Markup Language (VoiceXML) version 1.0. W3C Note, May 2000. See http://www.w3.org/TR/voicexml/.

W3C, XHTML 1.0: The Extensible HyperText Markup Language. W3C Recommendation, January 2000. See http://www.w3.org/TR/xhtml1/.

W3C, Web Services Description Language (WSDL) 1.1. W3C Note, March 2001. See http://www.w3.org/TR/wsdl.

W3C, Resource Description Framework. See http://www.w3.org/RDF.

WSIL, Web Services Inspection Language (WS-Inspection) 1.0: http: // www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html.

UNCORRECTED PROOF