Software Downloading Solutions for Mobile Value-Added Service Provision

Ouahiba Fouial[1], Nikos Houssos[2], Nadia Boukhatem[1]

[1] Ecole Nationale Supérieure des Télécommunications, Département Informatique et Réseaux, 46 Rue Barrault, Paris, France, email: fouial@inf.enst.fr, boukhate@infres.enst.fr

[2] Dept. of Informatics, University of Athens, TYPA Buildings, Panepistimioupolis, Ilisia, Athens, Greece, email: nhoussos@di.uoa.gr

Abstract: *Supplying mobile users with downloadable value-added services represents a great challenge for mobile telecommunication companies which, in an effort to enhance their service offerings, seek new forms of revenue other than voice telephony.*
This paper describes a technical solution for the implementation of a software downloading platform for service provision. This solution is based on Java and CORBA technologies which allow to ensure basic platform functionality. In addition, a mobile agent-based approach is investigated to complement this solution by facilitating and optimising the development of specific tasks. Two agent-based services are proposed: Service Downloading and Service Discovery.

Introduction

The customer's expectation of future wireless communication systems is access to multimedia services anywhere and anytime using a large variety of mobile devices. These multimedia data services will be provided either by the Mobile Operators (MO) themselves or offered through independent Value Added Service Providers (VASP) and will constitute an additional major source of revenue besides voice telephony.
In this context and within the frame of MOBIVAS[1] project, our aim is to define a distributed application platform supporting the provision of downloadable value-added services to mobile subscribers in a seamless and easy-to-use manner [1].
In this paper, we discuss possible technical tools able to cope with the issues of software downloading and propose a combined solution based on Java, CORBA and Mobile Agents (MAs).
This paper is organised as follows. Section 2 presents the context of our work and particularly the network infrastructure on which the software downloading platform will be deployed. Section 3 presents some existing technologies able to fulfil software downloading requirements and describes our proposed platform. Section 4 provides two scenarios, which demonstrate the benefits of mobile agents for service provision. Finally, section 5 summarises the key points of the paper.

Network Architecture for VAS Provision

The network architecture intended to support the distributed software platform consists of three basic entities: the mobile terminal, the Network Operator (NO), and the VASPs (which offer their services by connecting their network infrastructure to the network operator) (Figure 1).
Upon existing access and core network architectures, additional software modules are introduced, such as the VAS Manager (VASM), the Access Network Added Intelligence (ANAI), and the Charging-Accounting and Billing (CAB) system.
The VASM is mainly responsible for the management of the provided service list. It should present to the users a complete and updated list of all available services. In addition, it should cope with the co-ordination of functions required for the seamless and consistent VAS downloading to the mobile users.

[1] MOBIVAS: Downloadable MOBIle Value-Added Services through Software Radio & Switching Integrated Platforms, is an European IST project. URL at : http://mobivas.cnl.di.uoa.gr

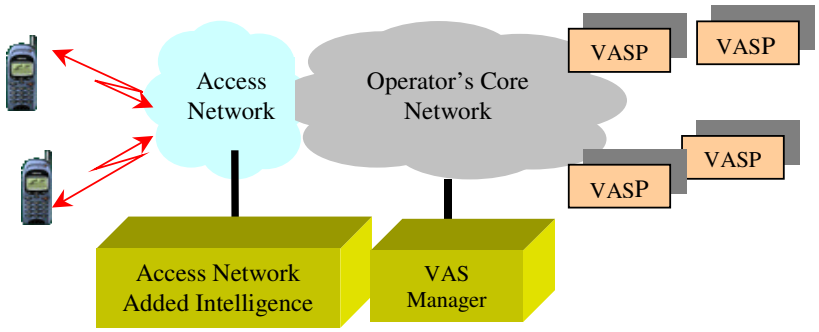

Figure 1 – Network architecture for service provision

The ANAI supports a terminal classification and negotiation procedure between the mobile terminal and the VASM to acquire the terminal capabilities and user profile.
The CAB system deals with the charging, billing and accounting operations induced by a service downloading procedure.
During the negotiation procedure, the VASM identifies all the relevant parameters for the service downloading (e.g. terminal capabilities and user profile) and provides to the mobile user a Lookup Service containing the list of all available services that are applicable to its terminal capabilities.
After the user selects a VAS to use, the corresponding service components are downloaded from the VASP to the mobile terminal and the charging process is initiated and controlled by the CAB system.

Software Downloading Technologies

To achieve the overall desired functionality of the software downloading platform, two main requirements should be met: Openness and Software downloading.
Openness concerns the interoperability issues which could arise between the different components of the platform. The network operator and the VASPs should be able to freely implement their services using different software products and easily integrate new services to the VAS platform.
The software downloading operation consists of moving the provided services (or downloadable service parts) from the VASPs to the mobile terminal where they can be executed. Thus, the mobile terminal should offer an environment able to support service downloading facilities as well as a service execution capability.
The first step towards the definition of a downloading environment for mobile terminals was the WAP (Wireless Application Protocol) standard [2] that enables Internet content and advanced services presented in the web to be accessed by wireless terminals. The main idea of WAP was to address the constraints of a wireless environment and adapt existing Internet technology to meet these constraints. However, WAP environment is limited; it does not support multimedia applications and service downloading.
To address this issue, the trend is towards the introduction of an enhanced execution environment within the terminal. Java and MExE (Mobile Station Application Execution Environment) [3] environments seem to be two interesting candidates. MExE's aim is to provide a comprehensive and standardised environment for mobile phones to execute applications. However, these applications are network operator or service provider specific. The idea of introducing Java technology in mobile terminals allows the transformation of a terminal into an open platform which benefits from the interesting advantages offered by Java (the standard interfaces (APIs) and the dynamic behaviour).
Many initiatives aiming to define Java APIs for constrained environments such as: PersonalJava [4], Embedded Java [5], and JavaPhone [6] have been proposed.

Java for software downloading

The Java language offers interesting capabilities for software downloading such as Applets and RMI (Remote Method Invocation).

Applets [7] are programs dynamically loaded in the browser and executed in its context. However, they present a number of restrictions and constitute a very limited solution for software downloading. Applets restrictions concern their impossibility of writing and reading on local resources, opening network connections and reading machine characteristics.

RMI [8] is essentially an object-oriented RPC-like facility for network programming. It has several features, the most important is its ability to download the byte code of an object's class if the class is not defined in the receiver's virtual machine. The types and the behaviour of an object, previously available only on a simple virtual machine, can be transmitted to another, possibly remote, virtual machine.

Mobile agents for software downloading

Mobile Agent technology is another alternative which represents an interesting approach for software downloading.

A mobile agent can be defined as a piece of software that is able to migrate at runtime from one network node to another and seamlessly continue its execution there [9]. Java Applets can be seen as an example of a very limited mobile agent. Indeed, if a Java applet can be downloaded from any server and runs on client's system without having the ability to access all resources, a mobile agent is able to move from a network server to another while having access to all required resources to execute itself.

The benefit of using mobile agents for software downloading is due to the fact that the moving of code portions can be done transparently. Indeed, agent platforms provide transparent mechanisms for code mobility, which are generally encapsulated in a single primitive or method.

This capability can be used for a transparent moving of VASs between the user's mobile terminal and service providers.

Moreover, this capability eases the development of distributed applications by hiding from the developers the implementation details of migration mechanisms.

The mobile agent concept provides other advantages [10] which allow for efficient software downloading and flexible service provision.

One of these advantages is the customisation: mobile agents allow the automation of the software deployment, installation and maintenance procedures. This could be useful for service providers in maintaining and installing new services.

Another MA advantage, asynchronous interaction, is expressed by the user's ability to delegate tasks to agents and to reduce network traffic.

In the client/server model (such as RMI), an ongoing interaction between the client and the server requires an ongoing communication. Thus, the network is called upon during the whole client/server session to carry the messages exchanged between the client and the server. With the mobile agent paradigm, a client and a server can interact without using the network once the agent has been transferred.

In addition, the agent will interact with the servers on behalf of the user. Thus, once the user sends its agent, it does not need to stay connected to the network waiting for the request (or task) results. The agents can wait (on a given server) until the user reconnects to the network before returning the results to him.

This makes the agent concept well suited for mobile computing environments where the devices are often disconnected from the network and have low bandwidth at their disposal. By using mobile agents, the device is connected to the network only long enough to send the agent on its way and, later, to welcome it home, avoiding the transfer of multiple requests and responses across the low-bandwidth connection.

Java/CORBA based Platform for VAS Provision

A comprehensive distributed software platform supporting the provision of downloadable VAS to mobile subscribers should consist of components running on a variety of network equipment, from handheld devices with limited capabilities to powerful workstations. Moreover, different interfaces

between these components have different requirements. Thus, the development of an efficient and complete framework requires the use of a combination of available distributed software technologies.
In our model of VAS provision, the service implementation can be divided into a *movable* (downloadable) part and a *stationary* part. The latter is typically a server that runs at the service provider premises and implements the basic functionality of the service. The former would typically be a lightweight client, consisting of a graphical user interface (GUI) that enables the user to access the service, as well as the necessary logic for communication with the stationary server. The movable part should be downloaded to the mobile terminal and executed there. This implies that it should be written in a machine- and operating system-independent language, so that it can be executed in mobile terminals from different manufacturers. We believe that Java is the most appropriate language for the development of downloadable clients, given its inherent portability and its other features facilitating downloading described in the previous section.
For server-side components a significant issue is openness. Each mobile operator or VASP should be able to implement the desired functionality using the programming language of his choice, while ensuring interoperability with the components developed by other operators and VASPs. CORBA [11] is a standard architecture for open distributed systems, which enables that, by allowing access to server objects (written in various programming languages) through open interfaces.
We propose that the VAS manager functionality is provided to the clients via CORBA objects. These objects are implemented in the network operator's programming language of choice and supported by any CORBA implementation and Object Transaction Monitor (OTM), installed in the NO server hosting the VAS manager. Clients (e.g. ANAIs, VASPs) need a CORBA module in order to communicate with the VAS manager.
For this communication they could use the lightweight ORB, called JavaIDL [12], included in a Java environment (A Java environment would anyway be present in all parts of the network), thus removing the need and cost for purchasing and installing a separate CORBA implementation. Java clients can also communicate with servers through the RMI-IIOP [13] extension of the Java language, which enables CORBA server access by Java clients using RMI semantics.
This approach provides several advantages, as it combines two complementary technologies, that is Java and CORBA. The use of CORBA in the server-side provides openness, multi-language support and scalability, as the number of server objects becomes very large. Various advanced services are available by CORBA for the development and maintenance of full-blown, scalable, extensible, sophisticated distributed applications. Moreover, access to CORBA servers through Java clients removes from the resource-constrained equipment of the network the burden of a CORBA installation. Similar platforms have been successfully developed for fixed network environments [14].
While the above client/server-based approach is adequate for the basic functionality of the platform, the use of mobile agents can complement this solution by facilitating and optimising the development of specific tasks. However, as agent middleware is resource demanding, the agent platform can be installed only in the core network and in case of specific terminal classes (PDAs, laptops) where there is realistic to have considerable computational and memory resources able to host the mobile agent software platform.
In the following section, two agent-based services: s*ervice downloading* and *service discovery* are presented.

Agent-based Service Provision

Software downloading
An agent-based scenario for software downloading is presented in the following:
In the MOBIVAS model for VAS provision, the VAS manager controls the downloading procedure, between the VASP and the MT, after the negotiation phase (during which the VASM is given information about the user profile and terminal capabilities) and the service selection phase (during which the user selects the desired service).
At the VASM premises, a mobile agent platform can provide access to two stationary agents namely the SPA (Service Provider Agent) and the TSA (Terminal Service downloading Agent).

The SPA allows the creation of mobile agents which are sent to the VASP to download services, handles the incoming mobile agents and interacts with them. This configuration based on stationary agents allows to limit the agents to access directly to the resources.
The TSA is responsible for the interaction between the VASM and the terminal for service downloading needs.
When receiving a service downloading request, the SPA creates a mobile agent called MPA (Mobile Provider Agent) which contains all the information able to identify the service at the VASP level. Once arrived to VASP, the agent first recovers the service code and then updates its itinerary. An itinerary is associated with each mobile agent; it is defined as a sequence of place addresses through which the agent passes to arrive to its final destination.
As mentioned above, an agent platform is required and deployed in the core network (VASM and VASP) and possibly in the terminal part if the latter have sufficient resources to host the platform.
In case of a resource constrained terminal (mobile phone, for example), the final destination is the VASM. Once arrived to the VASM, the MPA is taken in charge by the TSA, which uses the RMI capability to download the service to the mobile terminal.
The TSA handles the downloading between the VASM and the mobile terminal and provides several benefits: if the terminal is disconnected, the TSA stores the service in the VASM until the terminal reconnects to the network and the downloading process is triggered.
In case of a PDA or a Laptop, the itinerary final destination is the user terminal. Thus, the MPA moves the code from the VASP to the mobile terminal.
In this case, TSA capabilities can also be used if the connection between the VASM and the terminal is broken for any reason.
This scenario presents several benefits: first, an ongoing interaction between the VASM and the VASP is not needed since the MA transports all the parameters required for service downloading. In addition, the VASM delegates the downloading task to the MA and it does not have to wait for the VASP responses. This mechanism enables to decrease both the load at the VASM and the response time in case of large number of service downloading requests are simultaneously sent to the VASM.

Service Discovery while roaming

Another field where mobile agents can play a significant role is intelligent information retrieval. A relevant challenging issue in mobile VAS provision is enabling the user to locate the desired service between possibly thousands of VASs available from the operator. In the case of roaming, this problem becomes even more complex. The subscriber may need services not available from its home operator (e.g. specific information for the country or city he is currently located). Moreover, it is highly desirable for him to find the VASs more suited to his needs (in terms of QoS, price, content) among the services offered by not just the serving network, but a group of operators (e.g. all the operators in the country where he is currently roaming).
The proposed agent-based scenario for addressing this issue is presented in the following.
When a roaming user wishes to access the discovery service, he can formulate a request through an appropriate Graphical User Interface (GUI). This GUI includes an advanced search facility, enabling the specification of detailed requirements, like service description, desirable QoS and price. After the search form is completed, it can be submitted via an asynchronous (non-blocking) message to the serving VASM of which the SPA creates a mobile agent called SDA (Service Discovery Agent).
This SDA moves consecutively to a number of VASMs of different networks[2] and searches for services matching the user request among the VASs offered by the corresponding operators. In the simplest case only the home and serving network VASMs will be visited. The agent possesses the intelligence required to perform negotiation with the each SPA of visited VASMs and retrieve the best matches to the user's needs. This negotiation could include search refining and intelligent service filtering, according to user preferences and terminal capabilities. Service data is exchanged in the form of XML documents[3]. After concluding the search (or potentially only after finding a satisfactory match) the agent returns asynchronously to the user (possibly via the VASM) the results, in the form of a list of service descriptions and the corresponding links for service access.

[2] A prior contractual agreement between mobile operators is required to make this possible.
[3] A universal XML Document Type Definition (DTD) for VAS description should be defined for that purpose.

The above scenario is attractive for several reasons. The most important is that the required task is performed asynchronously. There is no need to maintain a network connection over the unreliable and costly wireless link for the whole duration of the task. Alternatively, this can be achieved by asynchronous messaging (e.g. message queues). However, in that case a reasonable degree of interactivity would require a large number of messages, resulting in significant task completion delay, as well as increased network traffic. In the mobile agent case, the agent contains the necessary logic information for the interaction with the various VASMs. Moreover, application development in the messaging case is far more complicated, because the programmer has to deal with any possible communication problems, while with mobile agents, the supporting middleware platform hides the communication complexity from the application developer.

Summary and Conclusion

The development of an efficient software downloading framework requires the combination of various distributed software downloading technologies.
The solution presented in this paper is based on Java and CORBA and uses mobile agents to optimise specific functions of the platform.
This combined solution seems to be quite appropriate for the software downloading platform. It provides all the necessary and desired features for the provision of value-added services (code downloading and openness). The use of CORBA in the server-side provides scalability and Java in the client-side avoids to burden the resource-constrained equipment of the network. On the other hand, the use of mobile agents leads to more efficient and flexible implementations of certain platform functionality.

Acknowledgements

This work has been performed in the framework of the project IST MOBIVAS, which is partly funded by the European Community. The Authors would like to acknowledge the contributions of their colleagues from Thomson-CSF Communications, Hellenic Telecommunications Organisation, NEC Europe LTD, University of Athens, Ecole Nationale Supérieure des Télécommunications, Technical University of Berlin, OTE Consulting, UNIS, IDATE, Innovators.

Bibliography

[1] N. Alonistioti et al., An application platform for downloadable VASs provision to mobile users, IST Mobile Communications Summit 2000.
[2] WAP Forum: www.wapforum.org
[3] Mobile Station Execution Environment (MExE), 3GPP Stage 2 Specification, 23.057.
[4] PersonalJava http://java.sun.com/products/personaljava/
[5] Embedded Java http://java.sun.com/products/embeddedjava/index.html
[6] JavaPhone API http://java.sun.com/products/javaphone/
[7] Applets http://java.sun.com/docs/books/tutorial/listofapplets.html
[8] Java™ Remote Method Invocation Specification. Revision 1.7.Java™ 2 SDK, Standard Edition, v 1.3.0, , December 1999. http://java.sun.com/j2se/1.3/docs/guide/rmi/spec/rmiTOC.html
[9] V. A. Pham, A. Karmouch, Mobile Software Agents: An Overview, IEEE Communications Magazine, July 1998
[10] S. Krause, T. Magedanz, Mobile Service Agents enabling Intelligence on Demand in Telecommunications, Proc. IEEE GLOBCOM'96, 1996
[11] CORBA: www.omg.org
[12] Java IDL Documentation, http://java.sun.com/products/jdk/1.2/docs/guide/idl/index.html.
[13] Java RMI-IIOP Documentation, http://java.sun.com/products/rmi-iiop/index.html.
[14] S.Gessler et al, An Overview of the I²N Architecture, to be presented in the annual Internet Society Conference, INET2000, July 2000.
[15] Grasshopper: http://www.ikv.de/products/grasshopper.html