Simple Mobile Services for IMS

Andrea Polidoro, Stefano Salsano, Giovanni Bartolomeo
Dept. Electronic Engineering, University of Rome "Tor Vergata"
{andrea.polidoro, stefano.salsano, giovanni.bartolomeo}@uniroma2.it

Abstract

In this paper we present an open source platform for mobile services execution and creation that has been ported to work into the IMS architecture. The platform has been developed in the context of the Simple Mobile Services research project and it includes a mobile client called MOVE (Mobile Open and Very Easy) developed using the Java 2 Micro Edition (J2ME) platform. The paper describes the features of the service execution/creation platforms, then it shortly introduces the IMS (IP Multimedia Subsystem) , finally it describes the process of porting the Simple Mobile Services architectural elements into the IMS architecture

1. Introduction

Considering the huge penetration of mobile phones, mobile services have not yet reached their potential (end expected) market success. Among the reasons, there is the fact that mobile services are often difficult to use and to configure, difficult to find, difficult to develop and deploy. The "Simple Mobile Services" project has addressed these issues with the design and development of an open source platform for mobile service execution and creation.

The IP Multimedia Subsystem (IMS) architecture has been standardized by 3GPP. Using IMS, the operators can offer a potentially unlimited set of services to their customers. On the other hand, there are still some issues with the vision of IMS: there is no clear idea of a killer application for IMS, it is not clear how services will be provided, who will be able to develop services, if and how the operator will "open" their IMS platform to other developers. There is also a "chicken and egg" problem, as services/applications are missing because IMS platforms are not still in place, and IMS platform may not be deployed until there are clear ideas of "killing applications" that IMS can provide..

In this paper we describe how we have ported the Simple Mobile Services platforms into the IMS architecture. We aimed to demonstrate that the services/solutions that we have developed in the SMS project can be run within an IMS scenario, using the IMS technology and approach. We have mostly based our implementation on an Open Source approach. In fact, the SMS service execution platform is composed of:

- an open source client for Mobile devices (MOVE) [3] which uses J2ME technology,
- an open source communication middleware ("SMILE" [4]) based on SIP (we used the open source mjsip stack [6])
- a set of "server side" components that can be open source or closed source

As IMS platform, we have used the open source "OpenIMScore" [9]. The MOVE client is developed using Java 2 Micro Edition (J2ME) assuming the CLDC configuration so that is portable across a large set of mobile devices. We have extended the MOVE client to support IMS and we have released the result using an open source licence. To the best of our knowledge, this client is the only IMS client that runs on the J2ME and it is open source.

2. The MOVE Application

The Simple Mobile Service architecture relies on applications running on mobile devices, combining local and remote "components".

MOVE (Mobile Open & Very Easy) is an example of a mobile application exploiting the SMS architecture. Implemented as a Java 2 Micro Edition MIDlet, MOVE is a "service browser" that allows users to access the required services among the available ones. Services are implemented by using several components, which may reside in the local MOVE application or may be located in a remote server and communicate through a lightweight middleware as described in Section 0.

Services implemented for MOVE up to now include:

- MEMs (Mobile Electronic Memos) messaging and handling. MEMs are electronic notes containing a structured set of attributes associated with a specific class of information and can be used by humans and

applications to exchange information, for example related to a location, a person, a service, the status of an ongoing activity, etc. Figure 1-7 shows a possible use of a MEM describing a restaurant. Users "capture" MEMs from the environment or from other services, store them for future use (Figure 1-2 to Figure 1-4), share them with other users (Figure 1-8) and send them as input to other components (like the maps/navigation component, Figure 1-5). MEMs allow to drastically reduce the amount of information to be entered manually by users, which is a key feature for mobile services. The MEM concept [2] has been introduced by the SMS project and is considered an important enabler for simplifying the usage of mobile services.

- A maps/navigation application for outdoor environments. This component (Figure 1-5) can show street maps, find addresses and businesses, connect to an external GPS receiver via Bluetooth to get the current device position and show the route towards destinations. It interfaces with a proxy, which provides an abstraction layer for a generic maps/navigation service, and makes it possible to use different service providers, such as Google Maps or MSN Maps & Directions.
- A navigation application for indoor environments, which can show maps of a given "local" area like a shopping mall, an airport, a campus, etc. providing walking directions to points of interest. It may either be connected to an indoor positioning system, like an indoor localization system based on Zigbee tags, or use other means to acquire the user location, including visual tags recognition and manual user input.

Figure 1: Screenshots from the MOVE application

- A "Find my Friends" service, which provides information about the location of friends and it is fully integrated with the outdoor and indoor maps/navigation components, so that friends' positions can be shown on the maps and directions can be asked on how to reach a friend.

Among other available components we quote a weather information service, a train schedule and departure/arrivals information services, a proxy to the well-known online photo sharing application Flickr.com; These services have been adapted to the mobile environment not only by performing modifications to their content, as usually done by existing "mobile browsers"; rather, their interaction paradigm has been changed to better fit the users' experience over their mobile devices. For instance, data are pre-fetched so that the latest information is always available, minimizing the user interaction time. Likewise, manual input of information by users is drastically limited by exploiting context (location, current activity, time, etc.) and/or user profile information, made available by MOVE.

The platform is designed to offer a richer set of services with respect to the ones listed above. Ongoing work includes for example the implementation of other components such as: Semacode to scan visual tags, Bluetooth access, instant messaging, RSS reader, and a set of proxies to other services available on the web.

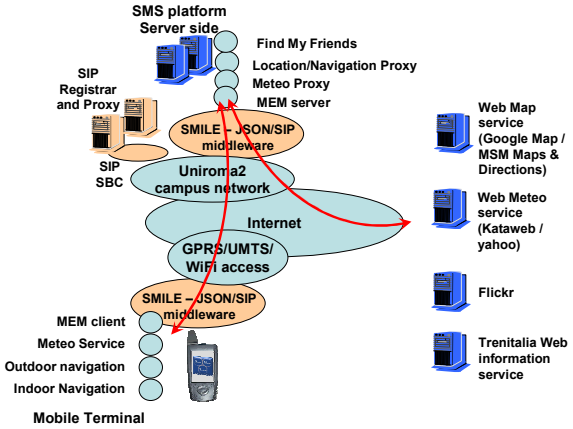

Figure 2: Test-bed scenario

We stress that the proposed platform is, to the best of our knowledge, the only one based on open-specifications, offering an open-source framework able to put together services and applications in a seamless and easy way. We also stress the novelty of the MEMs concept. MEMs are easily shared across different components and can be seen as tools to simplify the user interaction and as a "glue" that links services together .

The project has implemented a real test-bed whose architecture is depicted in Figure 2. Components and services implemented up to now, together with a rich documentation, are available in [3].

3. The SMILE Framework

SMILE ([4], see also [3]) is an abstraction layer written in Java which supplies a simple and unified framework for developing distributed, component-based applications. The components that need to interact are implemented as "SMILE peers". SMILE features an application-level peer-to-peer communication paradigm, which allows each peer to communicate with other peers either by sending asynchronous messages or by using synchronous remote procedure calls, at their convenience. In addition, it offers support for peer lifecycle as well as registration and lookup facilitations which may be exploited by any peer to find other peers.

Being an abstraction layer, SMILE is not a self-contained middleware and needs a "binding" to an underlying mechanism in order to implement actual communications. Components can therefore be developed using the SMILE framework, irrespectively of the concrete mechanisms that will be used for the communication. This way, applications maintain their portability across different middleware platforms and devices.

Formerly, we have implemented SMILE bindings on RMI, CORBA and JXTA middleware, all targeted to desktop platforms (e.g. using JAVA Standard Edition). In contrast, the JSON/SIP binding developed in the SMS Service Execution Platform allows all applications coded for SMILE to run on fixed hosts as well as on mobile devices implementing a J2ME virtual machine, without any change in their SMILE-based interfaces. It is worthy to note that, despite there exist examples of middleware platforms ported to mobile devices, it often happens that the set of APIs that they offer is only a subset of the ones provided by the original platform. Thus, applications need to be manually adapted to run on the mobile version. Mainly, this is because the original middleware platform has been thought for a traditional desktop environment, and is not provided in the form of an abstraction layer, like SMILE.

In the following we explain how we can allow SMILE applications to work seamlessly in J2ME CLDC devices, using JSON over SIP as a binding. Basically, we needed to address two issues: i) implementing a suitable *serialization* mechanism; ii) solving the NATs and firewalls traversal problem.

As regards the first issue, a *serialization* mechanism is needed to transform the internal representation of objects into a stream of bytes that can be interpreted and reconverted at destination. In Java 2 Standard Edition (J2SE), such a mechanism is built in. On the contrary, J2ME does not have an automatic serialization mechanisms and normally the application developer has to implement her own serialization mechanism for each application. Our approach has been to design a "seamless" serialization mechanism, which is performed by the framework itself: as shown in **Figure 3**, we create Java data types starting from a WSDL definition of the interfaces. The instances of these data type have methods to automatically produce at runtime a corresponding serialized stream (marshalling), and to rebuild the original object from a received stream (unmarshalling). Even if this approach is suitable for any serialization format, we have chosen JSON, which is a lightweight framework.

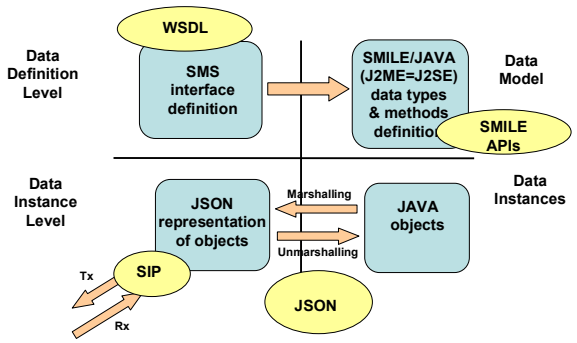

Figure 3: From interface definition to message instances

As regards the second issue, it is well known that NATs and firewalls do not allow peer to peer communications among mobile devices; for example, a peer within a "natted" network is not reachable from the outside world. In order to transport SMILE messages, we use the SIP protocol [5] and we resort to a known NAT traversal solution for SIP, based on the so called "Session Border Controller" element. The overall architecture of the JSON/SIP binding of SMILE over SIP is shown in Figure 4.

The SIP infrastructure is composed of the *Registrar and Proxy,* which maintains the mapping between SIP user agent identifiers and their IP addresses, and route calls to the recipients, and by the *Session Border Controller* (SBC) playing the role of intermediary for mobile clients behind NATs. This infrastructure allows the exchange of SIP "MESSAGEs" among mobile devices and between mobile devices and server side elements, even if they reside in different networks and behind NATs and firewalls. The SIP infrastructure elements and the SIP stack for both mobile devices and server side are based on the open source MjSip project [6]. The JSON/SIP binding of SMILE implements a fragmentation/defragmentation mechanism needed to send relatively large SMILE messages over SIP "MESSAGEs". It is based on a sliding window and avoids the fragmentation at IP level.

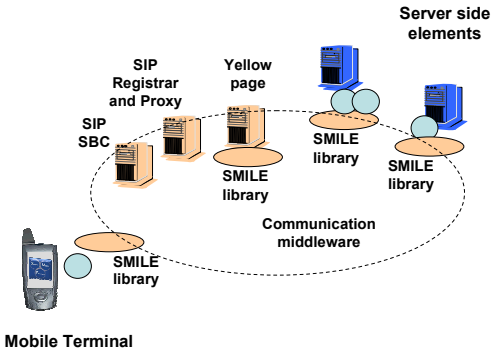

Figure 4: SMILE middleware and SIP elements

Figure 4 also shows the *Yellow Page* server, which allows SMILE peers to register/deregister their services and look for services offered by other peers. The SMILE abstraction layer offers this yellow page service to the applications, but the applications are not forced to use it (i.e., direct communication can be established without the support of the Yellow Page).

4. The IMS Architecture

The IMS [8] architecture is quite complex because it is composed of several entities and interfaces (as shown in Figure 5). However in this section we describe the main elements of the IMS network, that are the IMS Core Network elements and the Application Servers.

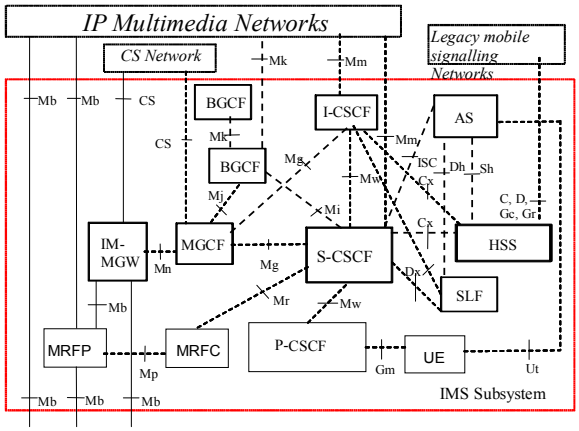

Figure 5: The IMS architecture

The IMS Core Network (CN) is composed of a database server (HSS) and three sip Server (CSCF).

4.1 HSS: Home Subscriber Server

The HSS (Home Subscriber Server) is the user's information repository. It stores, for each user, user profile information (e.g. the services subscribed from the user), security information (e.g. authentication and authorization) location information, and other items.
We can consider it as an evolution of the HLR (Home Location Register) in the GSM networks. The HSS implements the Diameter Protocol [7].

4.2 CSCF: Call-Session Control Function

The CSCFs are the SIP servers that manage and control the SIP requests received from the UE. Based of the role that the CSCF has in the IMS CN we can have three kind of CSCF:

1) The P-CSCF (Proxy CSCF) is the connection point between the UE (User Equipment) and the CN. According to SIP terminology we can define the P-CSCF as the outbound Proxy for the UE. The main functionalities of the P-CSCF include to establish and maintain a security association with the UE, forward its SIP request/response to the CN and generate the Call Details Records (CDR).
2) The I-CSCF (Interrogating CSCF) is the connection point between the local CN and the CN of different operators. Its main job is to forward requests/responses generated from local UE targeted to different CN domains. Also it manages the roaming situations and, during the registration procedure, assigns an S-CSCF server to the UE that needs to registers.
3) The S-CSCF (Serving CSCF) performs the session control services for the UE. During the registration procedure it acts as a SIP registrar/authentication server. It means that it retrieves the user's credentials stored in the HSS and uses them to challenge the UE. Also it maintains a session state as needed by the network operator for supporting the services.

4.3 AS: Application Server

An Application Server is a server that provides a specialized service. Typically in an IMS network there will be several ASs. Each AS can implement different technology (e.g. Java, Servlets, SIP CGI) in order to provide a user graphical interface but all ASs must implement a SIP interface with the S-CSCF named ISC (IMS Service Control). 3GGP defines three kinds of ASs:
- SIP AS: it is the native Application Server for IMS. It should be used for all the new services exclusively developed for IMS
- Open Service Access-Service Capability Server (OSA-SCS): it provides the gateway functionality to execute OSA services in the IMS.
- IMS-SSF (IP Multimedia Service Switching Function): it provides a gateway to legacy service networks that

implement CAMEL (Customized Applications for Mobile network Enhanced Logic) services.

5. Integration of MOVE/SMILE into the IMS Architecture

The integration of MOVE/SMILE in IMS architecture consisted in: 1) developing a MOVE client that acts as a IMS User Equipment; 2) developing SMS Server-side components in an IMS application server; 3) making adaptation to the SIP-based SMILE middleware so that it can work in the IMS environment.
Since the MOVE client and server can use SIP as binding for SMILE messages, the integration consist in the adaptation of the SIP stack inside the Move client and server in order to support the SIP extensions defined in [10] and in [11].
In the client side the canonical SIP registration has been adapted in order to perform a IMS registration.
In particular we have added:

- Private/public identity management,
- Support to Service Route Discovery in the Registration and the Subscribe Registration Package,
- Support to the P-Preferred Identity and the P-Access Network

In the server side the first step has been to group all the Simple Mobile Services server-side components (including the SMILE Yellow Pages) in a single Application Server. For this reason we defined additional fields in the "From" and "To" headers in order to distinguish among different SMILE peers running in the same Application Server.
Two new fields has been added in the From and To headers:
pType: it contains information about the kind of service implemented by the sender (receiver) SMILE peer
pName: it indicates the name of the sender (receiver) SMILE peer. This can be used to distinguish between more instance of the same service.
Thus, an example of a valid "From" header is the following:

```
From:
<sip:alice@neverland.net;pType=MEMreceiver, pName=istance0>;tag 123456;
```

In the Application Server (AS) implementation, the object taking care of delivering the SMILE message to the right SMILE peer is called Dispatcher (Figure 6).

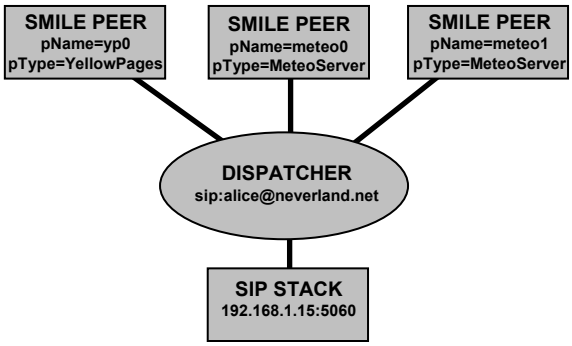

Figure 6 The role of a Dispatcher in the AS

This Dispatcher acts as SIP UA for the SIP stack, so it receives any SIP MESSAGE sent to its SIP URI. Whenever the Dispatcher receives a message it retrieves the SMILE message contained inside the SIP MESSAGE body, extracts the SMILE peer receiver from the pName and pType fields contained into the "To" header and forwards the message to the corresponding SMILE peer. In the opposite direction, when a SMILE peer wants to send a message to another peer, it specifies its own identification parameters (sender's pName and pType), the recipient identifications parameters (recipient's pName, pType and SIP URI) and delivers the SMILE message to be sent to the Dispatcher. Finally, the Dispatcher prepares the SIP MESSAGE and sends it using the SIP Stack.
As we run in an IMS environment, we have to perform some additional configurations in order to properly identify the AS not as a SIP User Agent (UA, as it was in the non-IMS implementation of SMILE) but as an IMS Application server. To that end, an Application Server entry has to be added in the HSS with the SIP URI of the AS, and one or more trigger criteria have to be added in the SCSF in order to forward the SIP requests to the AS. In this case the trigger criteria concerns the SIP Method (only the SIP MESSAGEs have to be forwarded to the AS) and the destination URI (only the messages with AS URI in the TO header have to be forwarded to AS). Finally, because the AS is not an UA the IMS registration procedure does not need to be performed; finally the S-CSCF has to be set as outbound proxy for the AS.

6. Open Source Applications and Testbed

In order to showcase the Simple Mobile Services features inside an IMS Network, we have implemented a test-bed as depicted in Figure 7.

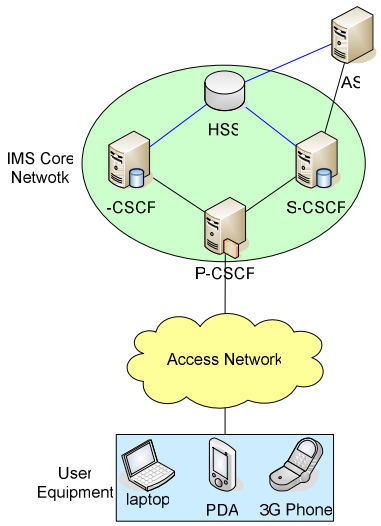

Figure 7 MOVE/SMILE IMS Testbed

For this test-bed we used 5 machines virtualized in one physical XEN machine and a set of smartphones (e.g. Nokia n70, N95, e61). Four of the five virtual machines are used to implements the IMS core network components (HSS, P, I, S-CSCF). Each of this machine is equipped with a Linux Debian Etch operative system. As software for the IMS core entities we used OSIMS, the Open Source IMS Core. This is an implementation of IMS CSCFs and a lightweight (HSS) based on open source software (SER SIP Express Router for the CSCF and MySQL for the HSS) developed by Fraunhofer FOKUS. The Application Server and the MOVE client are based on the MOVE client and the server side components developed in the SMS (Simple Mobile Service) project. Both of them use MjSIP [6], an open source SIP stack written in java by University of Parma and University of Rome Tor Vergata. Two different distributions of MjSIP (namely MjSIP-se and MjSIP-me) have been released to work respectively in J2SE and J2ME applications as well. We use MjSIP-me for the MOVE client and MjSIP-se for the Application Server. In both these implementations, the sip stack has been suitably modified in order to support IMS SIP extensions.

The developed open source component are available at [3], including in particular the IMS compatible SIP stack for J2ME and the IMS compatible mobile client.

7. Acknowledgements

This work has been performed in the context of IST Project “Simple Mobile Services” [1], partially funded by the EU. The author wishes to acknowledge Laila Aoufi, for her relevant contributions in implementation and proof of concepts.

8. References

[1] The “Simple Mobile Services” project, Project” IST 2006-034620, http://www.ist-sms.org

[2] R. Walker, G. Bartolomeo, N. Blefari-Melazzi, S. Salsano: "MEMs - Mobile Electronic Memos: efficient information capture and sharing for mobile users", WWRF #18, June 2007, Espoo, Finland.

[3] Simple Mobile Services – Platform documentation, http://netgroup.uniroma2.it/twiki/bin/view.cgi/Netgroup/SMSPlatformHome

[4] S. Salsano, G. Bartolomeo, N. Blefari-Melazzi, C. Trubiani, “SMILE, a Simple Middleware Independent LayEr for distributed mobile applications”, IEEE WCNC 2008, Las Vegas, USA, 31 March – 3 April, 2008.

[5] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. RFC 3261: SIP: Session Initiation Protocol, June 2002.

[6] MjSiP, open source Java implementation of SIP, http://www.mjsip.org/

[7] P. Calhoun, et al, “Diameter Base Protocol”, IETF RFC 3588, September 2003.

[8] 3rd Generation Partnership Project (3GPP), TS 23.228 IP Multimedia Subsystem (IMS); Stage 2 (Release 7) March 2008.

[9] The Open IMS Core Project: http://www.openimscore.org/

[10] M. Garcia-Martin, E. Henrikson, D. Mills January, RFC 3455 Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP,) January 2003

[11] D. Willis, B. Hoeneisen, RFC 3608 Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration, October 2003