Wireless SIP J2ME Application Development for Mobile Devices

Elthea T. Lakay, Johnson I. Agbinya**
Private Bag X17, University of the Western Cape, Bellville, 7535, RSA;
2001084@uwc.ac.za
***Faculty of Engineering, University of Technology, Sydney, NSW 2007, Australia;*
agbinya@eng.uts.edu.ua

Abstract—This paper illustrates and describes a methodology for developing UA for wireless devices, such as handsets and PDAs. The use of mobile phones and PDAs are increasing by the minute and thus the user demands and needs. Usage of SIP protocol as the signaling protocol is also increasing. Thus combining the them by creating applications comply to the demands and needs of the mobile phone and PDA users and industry. We describe the tools we used to develop applications. A simulation is shown to describe the tools we used.

Index Terms—Instant Messaging (IM), Java 2 Platform Micro Edition (J2ME), Session Initiation Protocol (SIP), Wireless Device

I. INTRODUCTION

In [1] and [2] we have studied the signaling problems associated with SIP. We showed that delay and security issues need to be adhered to, for SIP applications to match PSTN (SS7). In this paper we turn to the problem of creating fast SIP applications for PDA and small handsets with delay constraints. There are many sets of tools and libraries that developers can use to create rich applications for smart mobile devices like PDAs and handsets. The focus, however, will increasingly be on two main approaches [3]:

1. The .NET Compact Framework, which currently targets Microsoft Pocket PC and devices powered by Windows CE .NET
2. J2ME, which is not limited to Microsoft operating systems.

In this paper our focus is on the J2ME approach for developing SIP applications for mobile phones and PDAs. Later in the paper we outline how we used theJ2ME approach to develop SIP applications, how it is used, the difficulties we encountered, and the results.

The paper is structured as follows: Section II gives a brief view of SIP; Section III explains about J2ME, Section IV tells more about the SIP API for J2ME, Section V gives our view in Developing SIP J2ME Application for Mobile Devices, and in Section we conclude the paper.

II. SIP (SESSION INITIATION PROTOCOL)

SIP is an application-layer mobility-support protocol that can establish, modify, and terminate multimedia sessions [6]. SIP can also invite participants to already existing sessions. SIP transparently supports name mapping and redirection services, which supports personal mobility users that can maintain a single externally visible identifier regardless of their network location [6]. SIP can also be used to implement non-real-time services like Instant Messaging and Presence [9], [10].

Normally SIP supports five (5) facets of establishing and terminating multimedia communications:

1. **User location** – determination of the end system to be used for communication.
2. **User availability** – determination of the willingness of the called party to engage in communications.
3. **User capabilities** – determination of the median and media parameters to be used.
4. **Session setup** – establishment of session parameters at both called and calling party.
5. **Session management** – including transfer and termination of sessions, modifying session parameters, and invoking services.

These issues are covered in previous communications [1-2].

III. J2ME (JAVA 2 PLATFORM MICRO EDITION)

J2ME is used for building embedded applications for smart mobile devices [5]. J2ME configurations specify the minimum requirements for memory, Java language features, VM support and runtime libraries and do not include any optional components. It is available in two main configurations that incorporate a Virtual Machine (VM) and core API's:

1. Connect Limited Device Configuration (CLDC) for low-end, resource-constrained devices with limited connectivity. There are two options, called profiles.
   - The Mobile Information Device Profile (CLDC-MIDP) is widely used in hundreds of millions of phones today.
   - The Personal Digital Assistant Profile (CLDC-PDAP) is intended for future low-end PDAs that function primarily as PIMs. and

2. Connected Device Configuration (CDC) which is relatively new. It is intended for new, more sophisticated devices, including PDA devices. There are three profiles that build on each other and relate primarily to the increasing capabilities of the user interface. The most sophisticated of these is Personal Profile (CDC-PP) which equates to the capabilities in J2SE. It is also the natural competitor to the .NET Compact Framework.

There are also a whole set of optional packages that extend these profiles; they include Wireless Messaging API, Mobile Media API, J2ME RMI Optional Package, and the JDBC Optional Package for CDC Foundation Profile, as well as others still in the specification process, such as J2ME Web Services.

In our case we used the CLDC to develop both mobile phone applications and PDA applications. We specifically used the optional Wireless Messaging API package with the CLDC-MIDP and DLDC-PDAP profiles, since we develop Messaging applications.

1) Messaging

There are about 3 types of messaging in wireless environments [5]:

1. **Instant Messaging (IM):** In wireless environments, instant messaging usually means SMS messaging, because SMS bearer systems are typically used to implement the IM service. SMS bears provide instant messaging to the extent that messages are sent "immediately".
2. **Electronic Mail (e-mail):** E-mail is the transmission of arbitrary-length messages using a store-and-forward model. The exact capabilities of e-mail systems in wireless networks depend on implementation.
3. **Unified Messaging:** Unified messages is the concept of a single mechanism that unifies other messaging systems and gives users a single access point for all messaging services. A unified message system might have a user interface for both Web-based access and wireless device access.

We used the IM type. In wireless environment, IM usually means SMS messaging, because SMS bearer systems are typically used to implement the IM service. SMS bearers provide instant messaging to the extent that messages are sent immediately. This does not mean that the message get delivered immediately. Delivery time depends on system congestion, flow control, bandwidth constraints, and so forth, which can results in delays that are greater than those experienced in fixed network instant messaging technologies.

IV. SIP API FOR J2ME

SIP API for J2ME is a J2ME Optional Package that enables resource limited devices to send and receive SIP messages [7]. The API is designed to be a compact and generic SIP API, which provides SIP functionality in transaction level. The API is integrated into the Generic Connection Framework defined in Connected Limited Device Configuration (CLDC).

SIP API for J2ME can be used with the many J2ME profiles. The minimum platform required for this API is the J2ME CLDC v1.0. The API can also be used with the J2ME Connected Device Configuration (CDC). Figure 1 shows the simplified class diagram of the API, relations between classes, inheritance from javax.microedition.Connection and the relation to javax.microedition.Connector.

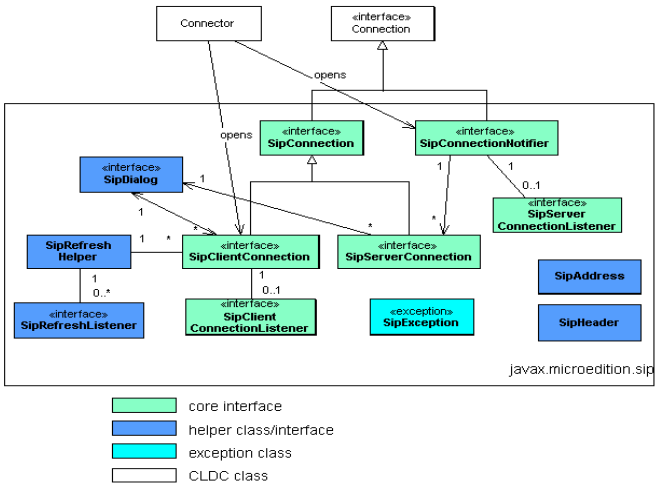

Fig.1. Simplified class diagram of the SIP API for J2ME [7]

The SIP API is used by the applications to implement SIP User Agents (UA) functionality. Figure 2 illustrates this scenario.

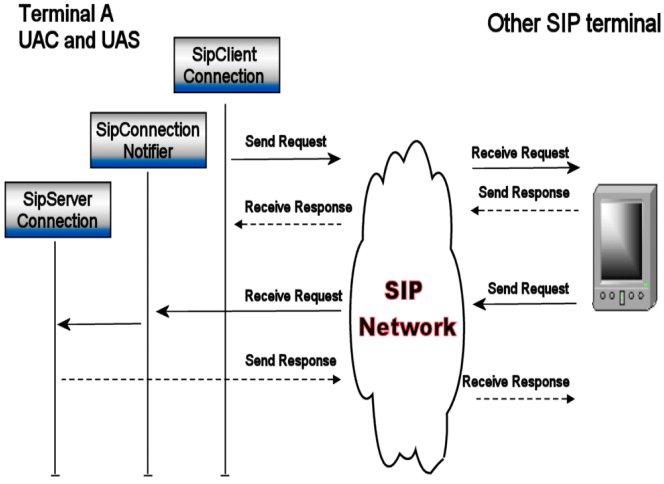

Fig. 2. SIP UA functionality, consisting of both client and server. [7]

A. CLDC (Connected Limited Device Configuration)

The main goal of the CLDC Specification is to standardize a highly portable, minimum footprint Java™ application development platform for resource-constrained, connected devices.

Devices that might be supported by this specification are:

- Cell phones
- Two-way pagers
- Personal Digital Assistants (PDAs)
- Organizers
- Home Appliances

In our case we only focused on the Cell phone and PDA.

This J2ME configuration specification defines the minimum required complement of Java technology components and libraries for small connected devices. Java language and virtual machine features, core libraries, security, input/output, and networking are the primary topics addressed by this specification.

CLDC is core technology that will be used as the basis for one or more profiles. A J2ME profile defines additional libraries are features for a particular vertical market, device category or industry.

SIP API for J2ME supports the following J2ME APIs:
- MIDP2.0
- CLDC1.1
- JSR180 (javax.microedition.sip)

B. J2ME Wireless Toolkit

The J2ME Wireless Toolkit (WTK) is a set of tools that makes it possible to create applications for mobile phones and other wireless devices. Although it is based on the Mobile Information Device Profile (MIDP) 2.0, the J2ME WTK also supports a handful of optional packages, making it a widely capable development toolkit.

1) The Tools in the Toolkit

The J2ME WTK has three main components:
- **KToolbar** automates many of the tasks involved in creating MIDP applications.
- The **emulator** is a simulated mobile phone. It is useful for testing MIDP applications.
- A collection of **utilities** provides other useful functionality including a text messaging console and cryptographic utilities.

KToolbar is the center of the toolkit. You can use it to build applications, launch the emulator, and start the utilities. Alternately, the emulator and utilities can be run by themselves, which is useful in many situations. If you want to demonstrate MIDP applications, for example, it's useful to run the emulator by it self.
The only additional tool you need is a text editor for source code editing.

2) Toolkit Features

The J2ME WTK supports the creation of MIDP applications with the following features:
- Building and packaging
- Running and monitoring
- MIDlet suite signing

C. Security Features

To send and receive messages (request/responses) using this API, applications MUST be granted a permission to perform the requested operations. The mechanisms for granting permission are implementation dependent.

1) Permission for Opening Connections

The MIDP2.0 specification defines a mechanism for granting permissions to use privileged features. This mechanism is based on a policy mechanism enforced in the platform implementation. The following permissions are defined for the JSR180 functionality, when deployed with a JSR118 MIDP2.0 implementation.

To open a connection, a MIDlet suite requires an appropriate permission to access the SIPConnection implementation. If the permission is not granted, then Connection.open methods MUST throw a SecurityException.

TABLE 1: PERMISSION GRANTED FOR EACH PROTOCOL

Permission	Protocol
Javax.microedition.io.Connector.sip	Sip
Javax.microedition.io.Connector.sips	sips

The table1 indicates the permission that must be granted for each protocol.

V. WIRELESS SIP J2ME APPLICATION DEVELOPMENT

To create the wireless SIP Applications for the Mobile Phone and PDA, we used the tools described in the above mentioned sections.

A. Mobile phone Applications

We started off with the simplest application for the mobile phone, the Chat, which we will explain later in this chapter. We then progressed to the IM application. We used Figure 2 scenario, thus both client and server in UA, to create the Chat User Agent and IM UA.

1) Development of Chat Application

We created a project in the WTK KToolbar, named Chat. Creating the Chat project, a folder, Chat, is created in the WTK application directory:
- C:\WTK22\apps\Chat

Within the Chat folder, the following folders are created automatically:
- Bin: where the JAR and JAD files of the application is stored
- Classes: where the class files of the compiled application is stored.
- Src: where the source code is stored

After the project is creates, we edit our source code in some editor and store it in the C:\WTK22\apps\Chat\src directory. This java file is then compiled and preverified, by building the Chat project in the WTK. The compiled files, class files, are stored in the C:\WTK22\apps\Chat\classes directory. Now we package the project, thus WTK compiles and preverifies the source files and bundles the java files and resource files into the MIDlet suite JAR files and MIDlet

suite descriptor. These MIDlet suites are stored in the C:\WTK22\apps\Chat\bin directory. The MIDlet suite is now ready to be installed into a real device or emulator. The project/application is now ready to be run and tested.

In our case we use the Nokia emulator, provided with the SIP API for J2ME, since the J2ME WTK emulator does not support SIP features. This emulator was stored in the C:\sipa1_0_1-ri-bin\midp\bin directory and its called sipa-midp.exe.

To run the Chat in the Nokia emulator, we used the command line:
C:\sipa1_0_1-ri-bin\midp\bin>sipa-midp –classpath /wtk22/apps/Chat/bin/Chat.jar Chat, whereby:

- **C:\sipa1_0_1-ri-bin\midp\bin** is the directory where the emulator is installed.
- **/wtk22/apps/Chat/bin** is the directory where the Chat package (JAR files, Chat.jar) is stored.
- **Chat** is the name of the project, created in the J2ME WTK.

2) Testing Chat Application

To use the application, a connection is required to another emulator or device, running on the same terminal or another terminal. When the connection is made, you are free to send and receive messages.

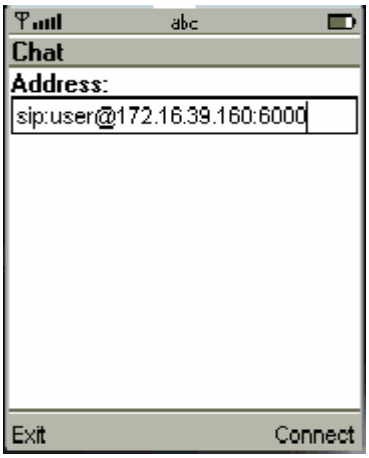

Fig.3. Connecting to another user.

In figure 3 you need to specify the address of the user you want to connect to on a specific terminal, and press connect. When connected, figure 4 will appear whereby you are able to send and receive messages. We set the connection to listen on port 6000 for received messages.

Fig.4. Sending and receiving messages

In figure 4 the Message box is there to type the message and send it to the user specified in figure 3. After typing the message and pressing send, a notification response appears, to verify if the message reached the destination or not.
"200 OK" means that the message reached it destination, as shown in the figure 4.
"MIDlet: exception Client transaction timeout" means that the message send to the specified user could not be delivered, since the user is unavailable. SIP tried to send the message for a number of times and when that time is over, it displays the message shown in figure 5. The trying time before displaying the error message is 19500 seconds. This time limit can be prolonged, but it is senseless, since it is a real-time application.

Fig.5. Error message when sending messages

3) Testing Results

We found it fairly easy to create the Chat application. We tested the application in a Local Area Network (LAN) and Wireless Network. In both scenarios the connection happens in real-time. We found that when sending the first message, it appears twice on the other user's terminal. The major problem with sending and receiving messages is that the user is not able to chat to multiple users at the same time, because of the capability of the mobile phone. This however does not mean that a multi-chat cannot be created, since all messages from different users go to the same open application.

The code size of the application (JAR file) is very small, 4.06KB, which can easily fit on mobile phone. This means the expected delay that this application signals in a network is neglible.

B. PDA Applications

To develop PDA applications in J2ME, you need a PDA optional package for the J2ME platform [9]. We are still in the process of developing applications for PDAs.

VI. Conclusion

In this paper we described and discussed some tools to develop SIP mobile applications. From the two main approaches, we used and described the J2ME approach. We used this approach since it's capable and efficient for use for mobile phones and PDAs.

To develop SIP J2ME applications, we used the SIP API for J2ME. We described how we used the SIP API for J2ME to develop applications for the mobile phone.

Future work involves developing SIP applications for the PDA and if time permits, developing a microserver for these developed SIP applications.

Acknowledgment

The authors would like to thank Telkom COE (Centre of Excellence) and the Department of Computer Science for giving them this opportunity.

References

[1] E.T. Lakay, J.I. Agbinya, "Communication Cost of SIP Signaling in Wireless Networks and Services", 12th International Conference on Telecommunications (ICT2005), Cape Town, South Africa, 03-06 May 2005.

[2] E.T. Lakay, J.I. Agbinya, "Security Issues in SIP Signaling in Wireless Networks and Services", Proc. IEEE International conference on Mobile Business, Sydney, Australia, 11-13 July 2005, pp. 639-642.

[3] B. Yoshimi, N. Sukaviriya, H Derby, B Carmeli, B. Bolam, J Elliott, J. Morgan, "Lessons Learned in Deploying a Wireless, Intranet Application on Mobile Devices", Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02), June 2002, pp.31

[4] D. Parsons, "Java Architectures for Mobilized Enterprise Systems", Proceedings of the 38th Hawaii International Conference on System Science (HICSS'05), January 2005, pp.298c

[5] V. Piroumian, "Wireless J2ME Platform Programming", Sun Microsystems Press A Prentice Hall Title, Release 1.0, March 2002

[6] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", RFC3261, June 2002

[7] SIP API for J2ME, JSR180, Version1.0.1, 2004

[8] J2ME Wireless Toolkit User Guide, Version2.2, Sun Microsystems Inc, October 2004

[9] Session Initiation Protocol (SIP) – Extension for Instant Messaging, RFC3428, December 2002

[10] Session Initiation Protocol (SIP) – Specific Event Notification, RFC3265, June 2002

[11] PDA Optional Package for the J2ME Platform – FileConnection Optional Package, JSR75, June 2004

Elthea T. Lakay received her BSc degree in Computer Science and Mathematics in 2002 and her BSc Honours degree in Computer Science in 2003, at the University of the Western Cape (UWC). Bellville, South Africa. She is studying towards her MSc degree in Computer Science, at UWC.

Dr. Johnson I. Agbinya holds an adjunct Professor in Computer Science at UWC. He holds a PhD in electronic communication engineering with expertise in cellular communication networks and software for communication systems. He is currently the Program Head of Engineering Practice Program Faculty of Engineering, the University of Technology Sydney, Australia.