Towards Developing Distributed Heterogeneous Mobile Phone Applications

Ali R. Mustafa Kattan[a], Rosni Abdullah[b], Rosalina Abdul Salam[c], Sureswaran Ramadass[d]

[a,b,c]School of Computer Science, Universiti Sains Malaysia
11800 USM, Penang, Malaysia
Tel:604-6533888 ext: 3610/2155/2170, Fax: 604-6573335
E-mail: [a]kattan@cs.usm.my, [b]rosni@cs.usm.my, [c]rosalina@cs.usm.my

[d]National Advanced IPv6 Center of Excellence(NAv6),
Level 6, School of Computer Science Building,
11800 USM, Penang, Malaysia.
Tel: 604-6533888 ext: 3004, Fax: 604-6533001
E-mail: sures@nav6.org

ABSTRACT

The advances in the mobile phone technology have enabled such devices to be programmed to run general-purpose applications using a special mobile edition of the Java programming language. Java is designed to be a heterogeneous programming language targeting different platforms. Such ability when coupled with the provision of high-speed mobile Internet access would open the door for a new breed of distributed mobile applications. This paper explores the limitations of this technology and addresses the consideration that must be taken when designing and developing such applications.

Keywords
Mobile Java, MIDlet Development, Distributed Applications.

1.0 INTRODUCTION

The wide spread of mobile cell-phones is far more ubiquitous when compared to the spread of PCs. Their processing power, as well as their storage capacity, has increased dramatically during the past few years (Knyziak and Winiecki 2003). The 'call' functionality, which is what the phone is about, became just one of many others functionalities that are equally important from a customer perspective. Taking and editing digital photos, watching live video, listening to music, are just few to mention. Applications can be added or removed depending on desired functions. The Java programming language became the common ground for developing applications for such phones (Xu 2006).

Fast Internet access via UMTS (3G), EDGE or WiFi technologies would become a standard low cost service provided to any mobile network subscriber. The relatively slow response time for the mobile applications that used to utilize the former CSD and GPRS technologies (Knyziak and Winiecki 2003) is something of the past. This would open the door for a new breed of mobile-based distributed applications that are to be integrated into larger existing computing infrastructures (Mock and Couturier 2005).

This paper explores the limitations as well as the consideration that must be perceived when designing and developing such applications. These were concluded based on actual tests done in comparison with their desktop counterparts. It is essential to have an idea about the Java mobile framework environment first in order to understand the nature of mobile application development.

This paper is organized as follows. Section II and III, will cover mobile Java framework. They also include highlights about the type of tests that need to be conducted to evaluate the capabilities that are considered relevant to distributed processing. The test model and the actual tests are presented in section IV and V respectively. Finally, the conclusions are in section VI.

2.0 PROGRAMMING MOBILE DEVICES

Sun Micro Systems Java programming language is one of the most popular languages used to program mobile devices. It is referred to as Java 2 Micro Edition or J2ME (lately known as Java ME). Basically, this is a cut-down version of the Java 2 Platform, Standard Edition (J2SE) that is tailored to suit mobile devices. As can be seen in Figure 1 (Xu 2006), Sun Micro Systems divides mobile devices into two categories; High-end representing PDAs and Low-end representing mobile phones and entry-level PDAs. The processing power of the former is usually 32-bit while the latter is limited to 16-bit which is the interest of this paper since they are more ubiquitous. Writing applications for mobile devices is totally different from writing applications for PCs (Mazlan 2006).

The framework is composed of a set of basic classes that are built into the mobile phone's firmware in addition to a set of optional packages that are loaded into the phone memory based on the application's needs.

JCP; Java Community Process (www.jcp.org) represents an alliance of participating members with most of the major mobile manufacturers and mobile service providers being involved. JCP is responsible for laying out the specifications for mobile Java. These are introduced in the form of JSRs; Java Specification Requests, to provide common implementation guidelines for mobile device manufactures and service vendors to undertake (Klingsheim, Moen et al. 2007). Such specifications are flexible to allow extension and promote compatibility. Despite of this, some manufactures have followed custom trends to add more functionality to their line of mobile devices. Unfortunately this would sometimes violate the promoted compatibility between different phone brands and might result in some unanticipated Java application bugs (Klingsheim, Moen et al. 2007).

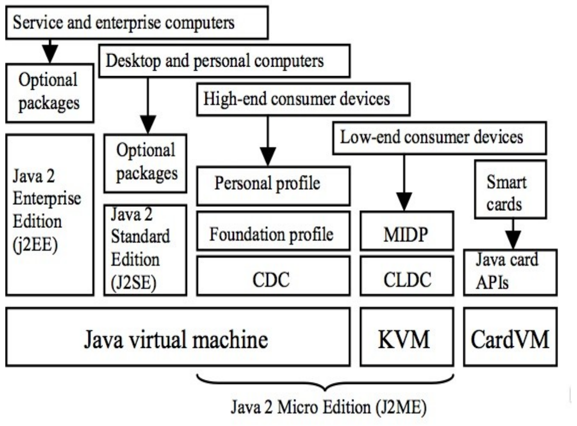
Service and enterprise computers
Desktop and personal computers
High-end consumer devices
Low-end consumer devices
Optional packages
Optional packages
Java 2 Enterprise Edition (j2EE)
Java 2 Standard Edition (J2SE)
Personal profile
Foundation profile
CDC
MIDP
CLDC
Smart cards
Java card APIs
Java virtual machine
KVM
CardVM
Java 2 Micro Edition (J2ME)

Figure 1: The different Java frameworks (Xu 2006)

2.1 Connected Limited Device Configuration

The configuration that defines small, mobile devices is known as the Connected, Limited Device Configuration (CLDC) (Sun Microsystems Inc. website). Examples of CLDC devices are mobile phones and pagers. These devices will have memory between 160 and 512 Kbytes and use the Kilobyte Virtual Machine (KVM) (Helal 2002) though such a memory based distinction is no longer valid. CLDC 1.1 (JSR 139) is the current version. CLDC 1.1 provides two basic packages for networking support:

- The java.io package, which provides classes for input and output through data streams. This includes reading of primitive data types streams and byte array streams.
- The javax.microedition.io, which provides classes for the Generic Connection framework. This includes creating connections (TCP based) and datagrams (UDP based).

Object Serialization is not supported and the created connections use blocking IO methods to achieve its functionality. Java RMI (Remote Method Invocation) is not supported under CLDC (Mock and Couturier 2005).

2.2 Mobile Information Device Profile

On top of the CLDC lies another set of classes, known as MIDP, that extend CLDC's functionality further (Klingsheim, Moen et al. 2007). This set is referred to as a profile. The Mobile Information Device Profile 2.0 or MIDP 2.0 (JSR 118), which is an enhancement over the former MIDP 1.0 (Sun Microsystems Inc. website), is currently the most commonly used profile in mobile phones.

Most of these enhancements address the security and privacy issues due to the added networking capabilities and the increased functionalities of the device (Klingsheim, Moen et al. 2007). The profile does not allow for security reasons dynamic class loading from sources different than its own JAR file (Mock and Couturier 2005).

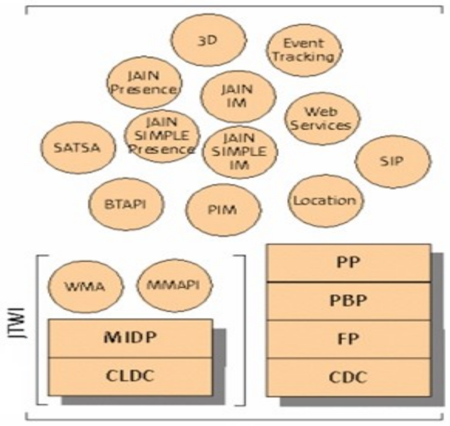

Figure 2: J2ME Optional Packages

2.3 Optional Packages

As can be seen in Figure 2, many optional packages can be added based on the intended application needs. These are also specified under JCP as JSRs (Klingsheim, Moen et al. 2007). These optional packages are tied to the provision of certain hardware features within the mobile device itself. For instance, the BTAPI package contains classes that enable the use and control of the device's Bluetooth feature if such a feature exists.

The same can be said about the other packages and the manufacturer should state clearly which of these are supported to facilitate application development and testing (Mazlan 2006). In addition to the optional packages, we found that the vendors sometimes would achieve extra functionality by providing their own customized packages as will be discussed later.

3.0 MIDLETS

A MIDlet is a J2ME mobile application. MIDlets are analogous to Java Applets known under the J2SE framework. The mobile phone has its own dedicated OS, namely the Application Management System (AMS). AMS is responsible for the loading, starting, pausing and destroying of MIDlets (Marejka 2005). Most of the recent mobile phones have a more complete and multi-threading capable OS like Symbian™ (Jode 2004).

3.1 MIDlet Lifecycle

In order to develop distributed Java based mobile applications it is essential to understand that MIDlets have different execution states (Marejka 2005).

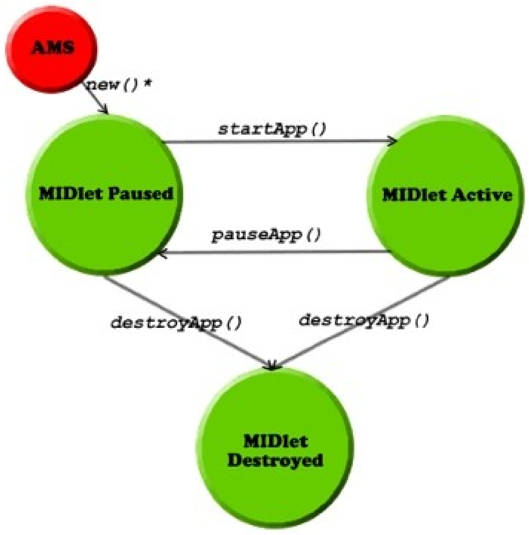

Figure 3: MIDlet Lifecycle

As can be seen in Figure 3, once the MIDlet files are installed in the phone's memory, the user can run the MIDlet by selecting it using a menu like GUI. The AMS would create an instance of this MIDlet and prepare it for execution. The MIDlet has three different states: Paused, Active, and Destroyed. All of these states are reflected by special methods within the MIDlet's code (Mock and Couturier 2005), (Helal 2002), (Marejka 2005). The Active state is where the MIDlet is doing its intended functionality. The paused state is the state where the MIDlet would be in the event of an incoming call or other high priority event that requires the MIDlet to pause. The MIDlet in such case would release its resources and wait till the high priority event is completed were by then it can ask the AMS to resume its functionality. Finally the Destroyed state is the state were the final house keeping is done to release any used resources and save any data prior to MIDlet termination. MIDlets can save persistent data on the phone memory using a system known as RMS (Record Management System) (Jode 2004).

Once the MIDlet instance is terminated, it seizes to exist from the working memory of the device. However it may keep the RMS saved data for use in the next run.

It is worth mentioning that recent mobile phones have a more capable operating system due to their higher processing power capability. Such OS would ignore the Paused state where the MIDlet may continue running in the background. The Nokia™ S40 series with its Symbian™ based OS is an example of this (Jode 2004).

3.2 MIDlet Development

Sun Micro Systems have provided a special development kit that makes use of the existing J2SE compiler to develop MIDlets. Java Wireless Toolkit for CLDC can be used to develop, test and debug mobile applications (Sun Microsystems Inc. website). It has a special set of emulators that will mimic a mobile environment.

The developer would use his/her preferred text-editor or integrate the toolkit with an IDE (Helal 2002) to edit the program code since it's not provided along with the kit.

The kit was used to develop some basic applications to test with. Sun's kit represents a generic platform to develop mobile applications without targeting a specific mobile brand. Java promotes the concept of 'write once run any were'. Unfortunately, this is not totally true when it comes to mobile Java applications. To be able to access the device's specific features and avoid compatibility issues that might exist between different mobile brands, special tailored versions of this kit are being offered by the device vendors (Klingsheim, Moen et al. 2007), (Helal 2002). These customized kits would include special packages that augment the original set. In addition, the emulators are extended to emulate actual commercial sets not just generic virtual emulators like those provided with Sun's toolkit. The developed MIDlets must be re-compiled and tested using those customized toolkits to avoid possible bugs (Klingsheim, Moen et al. 2007).

3.3 MIDlet Signing and Installation

MIDP 2.0 has introduced a new security model. In order to have trusted MIDlet suite the origin and integrity of the MIDlet must some how be authenticated. This is accomplished by having the MIDlet suite signed using a public key infrastructure (PKI). It uses the X.509 PKI, an ITU-T standard (Klingsheim, Moen et al. 2007).

Trusted MIDlet suites will be associated with a root certificate, which in turn is associated with a protection domain. The device vendor installs many of such root certificates on the device itself. The MIDlet suite should explicitly declare what permissions are needed. Such permissions must be a subset of the permissions given to the associated protection domain otherwise MIDlet suite installation will fail. The signing process is subjective to a fee by the root certificate party.

The MIDP 2.0 security model also provides the concept of protected API where access to those APIs is controlled by permissions. A Protection Domain is used to define a set of interaction modes and permissions, which grant access to an associated set of protected APIs.

An installed MIDlet suite is bound to one protection domain. MIDP 2.0 supports at least one protection domain; the untrusted domain. A set of protection domains supported by an implementation defines the security policy.

Signed MIDlets could acquire special privileges. Such privileges are not granted to "untrusted" MIDlets and user intervention may be needed to grant them access. This could become an inconvenient process and the MIDlet's functionality could be crippled if it's not granted the right permissions since user intervention is not always possible

4.0 DISTRIBUTED MOBILE APPLICATION MODEL

A client-server model is basically a distributed system where processes in the distributed system are divided into two (possibly overlapping) groups. The request-reply behavior is when the client is requests a service from a server by sending it a request and subsequently waiting for the server's reply (Tanenbaum and Steen 2002).

This model was adopted to develop a simple test application that would promote testing the distribution and networking functionality in the mobile devices and as seen in Figure 4.

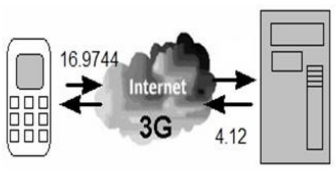

Figure 4: Test Application

In this application the client, which is supposed to be the mobile device in this case, would ask the server to provide a random (double) number. Upon receiving this number the mobile client is to calculate the 'Square' of it and then send it back to the server and so on. Care was taken to handle the application's "Paused" state such that the device would be able to store and retrieve its current connection state. The idea is to compare how the development of this application would go when targeting a mobile platform.

5.0 CONDUCTING TESTS

Or first goal was to see how convenient it was to build the mobile client MIDlet for the application discussed in the previous section. Our server application was hosted in a Windows 2003 server connected to the Internet using a global IP address. To build the client, we tested using three development kits (SDKs); Sun's, Nokia's and SonyEricsson's. Our MIDlet's code used the standard CLDC 1.1 and MIDP 2.0 avoiding any custom packages.

Since the environment lacks RMI and Object Serialization, we had to rely on building basic client-server sockets to achieve the mentioned functionality. J2ME relay's on blocking I/O methods and lacks the new non-blocking model available on the desktop version. Other than that the J2ME framework supported a wealth of classes and methods that are comparable to what is being provided on the desktop version J2SE. The development process on the three aforementioned SDKs was straight forward with no issues. We tested the client application using the included emulator application with each SDK. Although there are some considerable differences in terms of GUI appearance between the three emulators, the basic functionality is still the same.

The three MIDlets were uploaded to our server ready for download from a webpage using the mobile phone Internet browser. We used mobile phones from Nokia and SonyEricsson equipped with 3G Internet access. We faced issues in the applications installation process since our MIDlets were not signed. The installation and running of the MIDlet would require explicit user approval which must be granted. The same is true when the MIDlet tries to make an actual Internet connection.

The performance of the mobile application was acceptable in terms of speed, accuracy and the ability of handling network connections. However, our tests have also clearly showed how the 'behavior' of the mobile Java application would differ from one device to another due to different way of handling MIDlet's states and security measures by different vendors.

Unexpectedly, we have discovered another important issue that might hinder the proper running of our application. It seems that some mobile service providers would implement a NAT/Firewall solution for their subscribers' Internet access resulting in server-to-client communication problems.

Finally, we experimented with some 'customized' packages provided by the vendor. These packages address platform specific features or OS extra functionalities. Nokia S60 series SDK for example have included extra packages that address Nokia S60 series features and/or Symbian OS features. This would definitely whack out the sought compatibility since such MIDlets ran only on their respective vendor devices.

6.0 CONCLUSIONS

Java enabled mobile phones definitely have the potential for running diverse distributed applications. There are many programming limitations in the mobile Java version when compared to desktop Java. These limitations include the lack of RMI support, Object Serialization Support, and the support of only blocking I/O connections. Yet, and due to the increasing processing power and memory capacity of such mobile devices, those limitations can be compensated to a certain extent making mobile phones eligible candidates in any distributed computation that is part of a larger computer infrastructures.

Special care must be taken when developing such applications to ensure compatibility or at least portability across a wide range of the "said to be" compatible Java-enabled mobile devices due to different manufacturer implementations. Actual device testing is the best means to test for such issues.

Installing and running mobile Java applications that utilize Internet communication would require special permissions on the device itself, a bit of stringent security requirement. The running application is subject to be interrupted and paused by the device AMS. These issues must be well considered when designing and developing mobile based distributed applications.

REFERENCES

Helal, S. (2002). Pervasive Java. *Pervasive Computing* 1(1), 82 – 85.

Helal, S. (2002). Pervasive Java, Part II. *Pervasive_Computing* 1(2), 85-89.

Jode, M. d. (2004). *Programming Java 2 Micro Edition for Symbian OS: A developer's guide to MIDP 2.0*. Chichester, West Sussex, Wiley.

Klingsheim, A. N., V. Moen, et al. (2007). Challenges in Securing Networked J2ME Applications. *Computer* 40(2), 24-30.

Knyziak, T. and W. Winiecki (2003). The new prospects of distributed measurement systems using Java™ 2 Micro Edition mobile phone. *Proceedings of the Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications,* Lviv, Ukraine, 291-295.

Marejka, R. (2005). *MIDlet Life Cycle*. Retrieved December 15, 2008 from http://developers.sun.com/mobility/learn/midp/lifecycle/

Mazlan, M. A. (2006). Stress Test on J2ME Compatible Mobile Device. *Innovations in Information Technology*, Dubai, 1 – 5.

Mock, M. and S. Couturier (2005). Middleware - integration of small devices. *10th IEEE Conference on Emerging Technologies and Factory Automation*, 8 pp. - 814.

Sun Microsystems Inc. website. *Connected Limited Device Configuration (CLDC)*. Retrieved 10, Jan., 2009, from http://java.sun.com/products/cldc/.

Sun Microsystems Inc. website. *Mobile Information Device Profile (MIDP)*. Retrieved 10, Jan., 2009, from http://java.sun.com/products/midp/.

Sun Microsystems Inc. website. *Sun Java Wireless Toolkit for CLDC*. Retrieved 10, Jan., 2009, from http://java.sun.com/products/sjwtoolkit/.

Tanenbaum, A. S. and M. v. Steen (2002). *Distributed Systems Principles and Paradigms*. Singapore, Pearson Education.

Xu, C.-w. (2006). A Framework for Developing Wireless Mobile Online Applications. *5th IEEE/ACIS International Conference on Computer and Information Science, 2006 and 2006 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse,* Hawaii, 231-237.