Mobile IM as a Text Messaging Alternative Using Mobile Phone through the Development of Message Conveying System (MCS)

By

Azlan Fazly Mustaza

Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
INFORMATION & COMMUNICATION TECHNOLOGY
JUNE 2006

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Mobile IM as a Text Messaging Alternative Using Mobile Phone through the Development of Message Conveying System (MCS)

By

Azlan Fazly Mustaza

A project dissertation submitted to the
Information Technology Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
Bachelor of Technology (Hons)
INFORMATION & COMMUNICATION TECHNOLOGY

Approved by,

[illegible]
....

(MOHD HILMI HASAN)
UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
June 2006

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]
....

AZLAN FAZLY MUSTAZA

ABSTRACT

The development of *Message Conveying System (MCS)* as a mobile instant messenger application system is to provide a text messaging alternative using mobile phone. *MCS* is a mobile instant messenger application that is a part of global Jabber™ messaging networks. It means that you can chat and check *presence* of any user of any Jabber™ networks in the world just by using your mobile phone. *MCS* can run on any Java-enabled mobile phones that support General Packet Radio System (GPRS). Mobile phone users need *MCS* to bring the abilities of desktop *IM* into their mobile phones. *MCS* enables real-time communication between people. It also allows *presence* information to be available to others. This gives *MCS* an advantage over *SMS* because *SMS* does not provide the users' *presence* information to be available to others. *MCS* is also cheaper than SMS because its usage charges are based on the amount of data transferred through GPRS, not like *SMS'* pay per use charges. Apart from that, *MCS* accepts communication from different *IM* application that uses the Jabber XMPP protocol. It means that users who are using Jabber clients (e.g. GoogleTalk, Meebo, Spark, Exodus etc) can "talk" and see the *presence* information of an *MCS* user. *MCS* is seen to become the new main stream of text messaging because of the wide acceptance of *IM* among the people nowadays and the low usage charges that it provides which is way cheaper than the conventional *SMS*.

Keywords: Short Message Service (SMS), Instant Messaging (IM), presence, Message Conveying System (MCS)

ACKNOWLEDGEMENTS

Where do I begin? I thank the All Mighty Allah for his blessings and every door of opportunity He has opened for me throughout the period of completing this project. I thank my family, especially both of my parents for their support and encouragement.

This report should not have been finished without the help and guidance from my Final Year Project supervisor, Mr. Mohd Hilmi Hasan. His positive comments and critics have been vital in getting the best out of me for this project.

I also like to thank other UTP IT/IS lecturers for their views and suggestions during the presentations and evaluations. If it is not because of them, the project would have only been a mediocre or a failure. Not to forget to all my friends and peers throughout my 5 years at this university. The likes of Razaly, Arien, Wafa, Nazrul, Zeid, Suhaily, Firdaus and other people around have made my student life a worthwhile. The times spent with them will always be cherished.

Lastly, I thank again to all individuals who had given their support and thoughts on the completion of this project. May God bless us all, InsyaAllah.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

FONT CONVENTIONS

In this report, you will find different font formats for the *keywords*, `Java` syntaxes and source code snippets. This is to enhance readability of the report. The table below describes the font conventions used.

Key	Definition
Keywords (as stated in the Abstract)	Keywords (e.g *presence, IM*) are in *italic*
Java	Java syntax that is inside the text uses the font type `Courier New`
Source code	Source code snippets are all in smaller font size than the text of the report.

CHAPTER 1
INTRODUCTION

1.1 SMS & IM

Short Message Service (*SMS*) refers to sending and receiving text messages to and from mobile telephones. The text may be composed of words or numbers or may be an alphanumeric combination. *SMS* was created as part of the GSM Phase 1 standard. The first short message was sent in December 1992 from a PC to a mobile phone on the Vodafone GSM network in the U.K. Each short message ranges between 70–160 characters. *SMS* has a store-forward capability; this means sending messages is possible even when the recipient is not available. The user is notified when a message is waiting, as with voicemail.

Instant Messaging (*IM*) is an Internet-based protocol application that allows one-to-one communication between users employing a variety of devices. The most popular form of *IM* is chatting, where short, text-based messages are exchanged among computers.

Mobile *Instant Messaging* is the ability to engage in *IM* services from a mobile phone. It allows users to address messages to other users using an alias (or user name) and enabling the sender to know when his/her contacts are available.

1.2 Background

IM has been around for more than two decades. The first major player to enter the arena of *Instant Messaging* was AOL, which launched its own version of instant messenger with a component used for managing all the incoming and outgoing messages and the list of friends. This component is popularly known as buddy list. Soon, Microsoft and Yahoo! followed AOL's trail. As a result, MSN and Yahoo! messenger appeared on the market with a variety of impressive new services. In its early days, *Instant Messaging* uses were restricted to splashing messages on bulletin boards. Gradually, *Instant Messaging* became a major area of interest for youngsters. Society acknowledges *Instant Messaging* as the most common means by which people of varying age groups, especially youngsters, communicate with one another.

Until 1990, there was no significant change in the status of *Instant Messaging* from what it had been when initially conceived, mainly because *Instant Messaging* was not taken seriously till then. Subsequently, the corporate world changed its attitude toward *Instant Messaging*, thanks to the Internet revolution. With a sudden rise in the popularity of the Internet and the arrival of new techniques like voicemail, online transactions, and so on, the corporate world started taking *Instant Messaging* seriously. The Internet led not only to a substantial increase in the number of *Instant Messaging* users but to the realization of the potential of *Instant Messaging* and to earnest attempts to eliminate the limitations of *Instant Messaging* and exploit its possibilities fully.

1.3 Problem Statement

Text messaging has become a prominent part of our daily activities. The emergence of *SMS* and *IM* had contributed a lot into this matter. People send messages anytime and anywhere they go. *IM* is popular for sending messages using the computer while *SMS* is for sending messages using mobile phones. Currently, there have been many efforts to bring the functionalities of *IM* into mobile phones. The advantage of mobile *IM* is that messages are sent and received in real-time via mobile phones in the same way as fixed *IM* services, but without the need to be attached to a computer. Mobile *IM* is seen as a natural evolution of the popular *SMS* service.

To date, users have required an existing active *IM* account - such as that offered by AOL and Yahoo! - and a compatible handset (one with a pre-installed messaging client) running over a GPRS or 3GSM network. Mobile *IM* is available from some operators now, but it is not always possible to use *IM* services between different operators and different *IM* communities. However, this type of service has not yet been available to the users in Malaysia because the telecommunication operators in Malaysia have not taken the step to venture into this kind of service. This is because they fear that the introduction of *IM* will cost a much loss in profit to them as shown by Yuan that 40% of the profits for telecommunication operators come from *SMS* [Yuan, 2003]. Other study also states that Malaysian mobile telecommunication operators rely heavily on conventional *Short SMS* operations as it provides the bulk of its revenue [American.edu, 2004].

Malaysian users need a mobile *IM* that brings the abilities of desktop *IM* into their mobile phones. The mobile *IM* should allow communication between different operators and different *IM* communities. A person with the appropriate mobile phone that supports GPRS will have the ability to send and receive messages in real-time via mobile phones in the same way as fixed *IM* services, but without the need to be attached to a computer. In doing so, messages can be sent much faster, more efficient and the most important of all – cheaper than the conventional *SMS*.

1.4 Objective

The purpose of this project is to provide a text messaging alternative using mobile phone through the development of a mobile instant messenger application that:

i. Can run on Java-enabled and GPRS-supported mobile phone.
ii. Proves mobile *IM* as a cheaper text messaging communication than *SMS*.
iii. Allows communication between users from different *IM* application that uses the same protocol.

CHAPTER 2
LITERATURE REVIEW

2.1 Presence and IM

The advances of new technologies and the convergence of different communication media are constantly changing not only our means and modes of communication with other people, but the notion of connectivity itself. Rather that being online or offline, we can be 'connected' in many different ways and without directly interacting with technology itself. *Presence* awareness, facilitated by *Instant Messaging* applications, mobile phones, wireless handheld devices, location tracking and so on, can make someone reachable almost at any time.

Presence is becoming a key issue in the wired and wireless world. At the most basic level, *presence* awareness lets users know when other people in their contact list are online. However, the concept has expanded from the initial online/offline description to what we identify as rich *presence*. Thus, *presence* information can include more user details, such as availability, location, activity, device capability and other communication preferences, even expressed in more abstract terms, like 'mood' or 'intention'. A general notion of *presence* would answer the questions of Who (user), Where (location and device), When (preference and willingness), How, (device capability) and Why (information exchange, leisure, keeping in touch etc) [Chakraborty, 2002].

The most commonly used tools that facilitate *presence* awareness are the various *Instant Messaging* applications, though there is an increasing trend towards recognising that *IM* is itself just one (communication-oriented) of many facets of *presence* management. *Instant Messaging* is one of the fastest growing areas of the internet for the past few years that allows millions of users around the world to contact friends and colleagues in a convenient way, with more immediacy than e-mail and without the expense of a phone call.

An *IM* service can be either device-based or network-based. In the former case, the user needs to have a client application installed on a computer or other device. Contact lists, user information and preferences are then stored on the device. If a user wants to access the system from another device, for example a mobile phone or PDA, all the information must be set again by the user for every device. Any future changes (e.g. adding another contact to the contact list) would then need to be made manually for every device used to access the system. The advantages of a network-based system where all information is stored on a dedicated server and updated dynamically are evident. A client application still needs to be installed or embedded on every device used to access the system, but then all information need to be provided only once and all changes will be updated, so that all devices display the same information. However, this model has the trade-off of security risks and lack of privacy, particularly for corporate *IM* use. For wireless *IM* communication, a network-based system is highly recommended, partly because of low memory availability on handheld devices. A network-based system can also serve to provide translation mechanisms, allowing users with different device specifications (e.g screen size), capabilities and software to communicate with each other. While desktop *IM* users are quite used to having more than one *IM* application running, we expect that mobile users will be more willing to use only one client, due to the limited resources on mobile devices and since switching between clients is more complicated. For this reason, a server would need to provide gateways to other *IM* providers.

Most familiar today is computer-to-computer *Instant Messaging*, but *IM* is clearly moving to the mobile domain. Major desktop *IM* providers have already formed partnerships with mobile telecommunication operators to add *SMS* functionality to their systems. Currently *IM* is also implemented on mobile phones through WAP-based clients and also as a separate Java application for Java-enabled mobile phones.

Presence is the key feature that differentiates mobile *Instant Messaging* to the existing *SMS* messaging facility on mobile phones. Without *presence*, the user does not know of a person's availability or device status. *Presence* is a constantly evolving dynamic construct with a great potential for future telecommunication applications.

2.2 Today's Technology

With the introduction of the latest mobile phone technology and as *IM* moves to the mobile domain, one of the key functions is connectivity between the internet and the mobile world. Inter-working between *IM* on PCs and *IM* on mobile devices marks a true convergence [Nardi & Whittaker, 2000]. This inevitably affects usage patterns and can extend the functionality of *IM* to various directions. By adding the element of mobility and location to *IM*, another level of *presence* awareness is introduced. For example, users can find about other people's approximate location as well as availability. In this way, they can contact each other to meet when in vicinity. Other uses of this application, like multiplayer games, are also likely to emerge.

The telecommunications industry has high expectations from the integration of *IM* with mobile services. Apart from increasing revenue streams through advanced messaging services, wireless *presence* and *IM* are also beneficial for mobile commerce, business use and location based services (LBS) [Peretz, 2002].

With the addition of IP telephony capability, an *IM* application can become a major communications platform. When integrated with the phone, *IM* allows users to negotiate availability and avoid interruptive phone calls. In this way, an *IM* system can act as a personal communications portal, providing users with the ability to specify their communication preferences according to the device they are using or who they want or do not want to communicate with.

Presence technology and *Instant Messaging* integrated in the mobile domain will affect our daily communication patterns and behaviors, even more strongly than desktop *IM* has already done. *IM* is informal in nature, but provides great assistance for group collaboration. In professional and educational settings the impact of desktop *IM* has been significant so far: work coordination, meeting arrangements, quick exchange of information and a sense of being 'connected' with people at a distance are some of the most important facilities offered by *IM*. Mobile *Instant Messaging* can have an even greater effect, by providing more flexibility in time management and meeting arrangements as well as an 'always in touch' state, which can foster group interaction.

At the moment however, there are significant technological and economic constraints preventing *Presence* and *IM* from becoming truly revolutionary technologies, widely accepted by mobile device users. The problem originates from desktop *IM* and becomes more complicated when it comes to mobile phones. Unlike the internet, *IM* has not been based on open standards and therefore most users can only communicate with people using the same protocol. For *IM* to become a broader communications platform, the various systems need to be able to communicate on a standards basis, which means that major *IM* providers like AOL should open their systems to other providers. Despite valuable interoperability efforts, the issue is still the hottest topic in the industry. Mobile *Instant Messaging* on the other hand, requires the cooperation of mobile telecommunication operators for standards establishment. Since the technology is so new, it will be very interesting to see how it will evolve. In the near future, interoperability developments will determine the use and adaptation of mobile *presence* and *IM*.

2.3 Mobile Messaging Standards

2.3.1 SMS - EMS- MMS

Messaging on mobile phones is rapidly evolving from pure text messages (*SMS*) to messages that resemble multimedia presentations (MMS – Multimedia Messaging Service) and can include graphics, data, animations, audio clips, voice transmissions and short video sequences. In between *SMS* and MMS we have the Enhanced Messaging Service (EMS), which can contain combinations of text, simple images and melodies. The text of an EMS message can be formatted. EMS has added a degree of personalization to text messaging. MMS is definitely an important advancement over *SMS* and EMS, however new network infrastructure components and new handsets are required. The introduction of MMS has shown so far that it is not going to be a seamless customer experience due to serious interoperability problems [Tulloch, 2002]. Technical complexity put aside, it is still hard to achieve network interoperability for commercial reasons; operators use different tariff schemes to charge messaging services. There are concerns that users might not adopt MMS as they did with *SMS* because of interoperability problems as well as high costs.

2.3.2 IM and SMS – A hybrid technology

The advance of mobile messaging standards from *SMS* to MMS will not really influence the way *IM* itself currently works on mobile phones, apart from offering the ability to send audiovisual data to contacts.

The combination of *IM* with *SMS* resulted in a hybrid technology possessing attributes of both. Messaging patterns are quite similar, both *IM* and *SMS* messages are usually shorter than 100 characters with no attachments. Contrary to desktop *IM*, *SMS* enjoys global connectivity, like e-mail. *IM* benefits from the store-forward capability of *SMS* (i.e. messages are sent via an *SMS* centre), which allows users to send messages to recipients that might have their mobile phone switched off [Pulver.com, 2001]. The advantage of *SMS* as an established technology with a great user base and popularity in both Europe and Asia has created expectations in the industry that it will drive the uptake of mobile *IM*. Basic *IM* functionality (send/receive messages and contact list) can already become available on existing handsets and networks.

Therefore, it is nearly impossible to provide a cross-platform Java mobile application as developers need to take care to ensure that the GUI controls that they use work properly and somewhat consistently on each deployed platform. In addition, the user interface requirements must not exceed the various devices' capabilities and complexities that are unnecessary on some platforms have to be added in order to satisfy the requirements of others. The work of putting in the additional work and complexity to have one application that runs on multiple devices can be an endless process. This suggests that the "Write Once, Run Anywhere" of Java cannot be applied to mobile applications.

Currently in Japan, Java has successfully been deployed on NTTDoCoMo's i-mode platform. Interactive content on i-mode is more advanced and graphically interesting than most mobile content available in Europe (e.g. users can send full color animated messages with cartoon characters), [FunMail, 2002]. The J2ME technology is a driving force for content development on mobile phones, personal digital assistants and other handheld devices. Mobile *IM* can be implemented as a separate application on Java-enabled devices and in the future *IM* and *presence* information could be combined with other applications as well, for example multiplayer games.

2.6 Interoperability

Lack of interoperability is the greatest problem for both desktop and mobile *IM* [Woods, 2002]. In the desktop world, the greatest obstacles have been posed by major service providers, like AOL, who want to protect their user base. AOL has sidestepped *IM* standardization efforts from the Internet Engineering Task Force (IETF) and, citing privacy concerns, shut out rivals who figured out how to let their users access AOL's *IM* services [Ulfelder, 2001]. Interoperability is vital for the success of mobile *IM* and *Presence,* since users are unlikely going to pay for a service if they cannot communicate with people on other networks or using a different service provider. The value of an *IM* service to end users is dependent on the number of other users of the same service. A critical mass is required for early adoption, in order to fuel usage growth.

2.6.1 Billing – Related Problems

Unfortunately, most important interoperability constraints are not technical, but commercial in nature, particularly when considering the inter-working of mobile *IM* with existing desktop based *IM* services. Mobile operators have always been charging for messaging services, while desktop *IM* has been free for internet users, thus PC-to-SMS messaging becomes problematic since billing issues need to be resolved.

Many operators, particularly in the US, have formed partnerships with *IM* service providers. They are also likely to launch their own services in the future. The most important motives for this, according to "Wireless Instant Messaging" white paper [Wiral, 2001] are:

a) Larger share of revenues

b) Increased brand value

c) Greater control over service development

d) Security and reliability issues (Internet *IM* services have not proved secure enough for business use)

e) Leveraging of complementary services, such as location services by combining subscriber *presence* information with location information.

Alliances between mobile telecommunication operators and major internet *IM* providers usually operate on a dual model, without promoting interoperability standards for the rest of market.

2.6.2 Standards Bodies and Protocols

Several groups are currently working on establishing standard protocols for messaging interoperability, however, while their efforts do not necessarily contradict each other, those groups are not working together.

IETF – Internet Engineering Task Force

The Internet Engineering Task Force is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet [IETF, 2002]. Working groups within IETF focus on developing protocols for *Presence* and *IM.* The MMUSIC (Multiparty Multimedia Session Control) working group has developed the SIP (Session Initiation Protocol), a signaling protocol for Internet conferencing, telephony, *presence,* events notification and *Instant Messaging* [IETF, 2001]. Another group within IETF, the SIMPLE group, develops the SIP (Session Initiation Protocol) for *Instant Messaging* and *Presence* Leveraging Extensions. The SIP-based architecture of the SIMPLE protocol aims to integrate *presence* and *Instant Messaging* with traditional telephone communications and web conferencing. This protocol has acquired a lot of industry support so far, particularly from two of the largest software corporations, Microsoft and IBM. There is also the Instant Messaging and Presence Protocol (IMPP) group within IETF, working on protocols and data formats necessary to build an internet-scale end-user *presence* awareness, notification and *Instant Messaging* system [IMPP, 2002]. Finally, the most recently established group [XMPP, 2002] is the Extensible Messaging and Presence Protocol XMPP working group. XMPP is the XML-based core protocol of the Jabber *Instant Messaging* and Presence technology, an open source community initiative (see section **3.2.3** for a description of the Jabber project).

PAM Forum – Presence and Availability Management

The Presence and Availability Management (PAM) Forum is an independent, nonprofit consortium established to standardize the management and sharing of *presence* and availability information across multiple services and networks. The goal is to establish a standard for maintaining and publishing information about user identity, *presence* (including information like location, device state and communication capabilities) and availability. The PAM specification also provides a mechanism for privacy management, to allow users have control over their private information. The focus of the PAM forum is to develop and promote a *presence* and availability application programming interface (API) specification [PAM, 2002].

Wireless Village

The Wireless Village initiative was founded by Ericsson, Motorola and Nokia in April 2001 to define and promote a set of universal specifications for mobile *Instant Messaging* and *presence* services. The specifications concern the exchange of messages and *presence* information between mobile devices, mobile services and Internet-based *Instant Messaging* services. It is the only group with a clear focus on mobile *IM* and Presence; though the other groups also take wireless technologies into consideration, they keep a more general approach. The Wireless Village proposes a standard protocol for Instant Messaging and Presence Service (IMPS), which includes *presence* information management, *Instant Messaging*, group management and shared content [Ericsson, Motorola and Nokia, 2002]. Unlike the PAM Forum, which separates availability from the rest of *presence* information, in the Wireless Village specification, *presence* includes availability, as well as other information such as location, device capability, profile etc.

2.7 Summary

The research that had been made shows that mobile *presence* and *IM* are clearly very promising and have already attracted a great interest. The establishment of the 2.5G and 3G of mobile network technology will facilitate persistent ‘always-on’ connection with friends, family and colleagues through *IM*. Moreover, *presence* is becoming increasingly important and currently moving beyond *IM* to a variety of domains. Rapid technological convergence will continue and we can envision *presence* information becoming more ubiquitous and driving embedded software development beyond mobile phones. *Presence* will not only include the notion of user or device state, but almost anything can have a *presence* state, from the printer and the coffee machine to any work in progress in our computer.

CHAPTER 3
METHODOLOGY

The project follows the Waterfall Methodology where the planning, analysis, design, and implementation phases proceed in sequence from one phase to another. This chapter will describe the detailed explanations of each phases of the methodology.

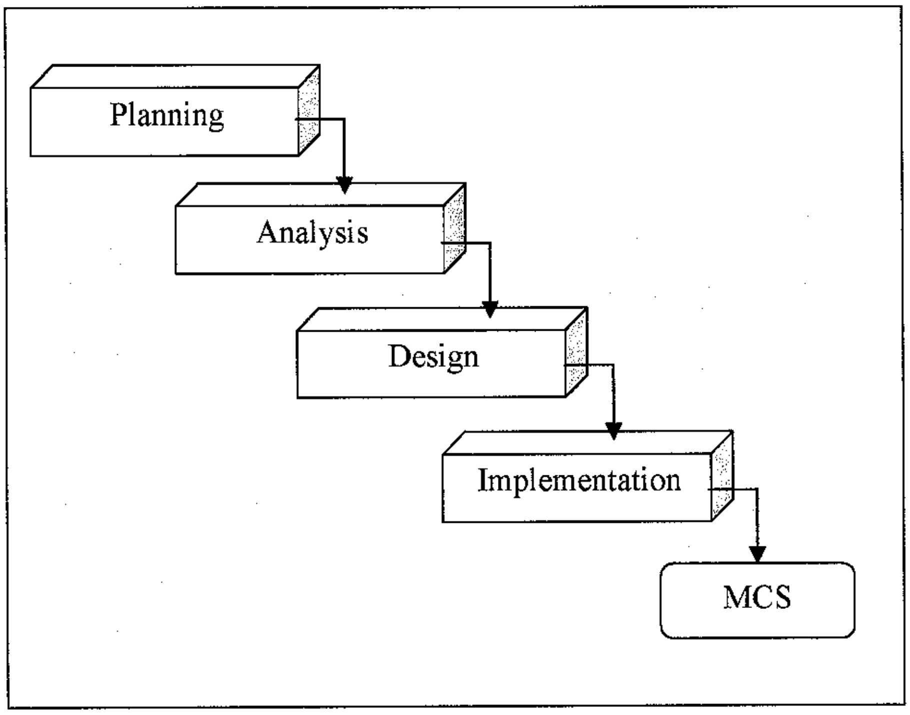


Figure 3.1: MCS Waterfall Methodology

3.1 Planning Phase

The planning phase of this project involves what to be produced and who are the target users of the end product. The platform and technology that can cater to the project objectives was chosen to develop the system.

The whole idea of this project is to develop a mobile *IM* that meets all of the project objectives. *MCS* is a mobile instant messenger that uses J2ME as its platform and utilizes the power of Jabber, an open-source *Instant Messaging* platform that uses open, XML-based protocols to create the standard functionality people expect of an *IM* system: one-to-one chat, multi-user chat, the ability to subscribe to someone else's *presence*, and so on. With the power of Java as a renowned platform independent software and Jabber XML capabilities, *MCS* is expected to run on all Java-enabled and GPRS-supported mobile phones.

3.2 Analysis Phase

During this phase, the current *SMS* rate in Malaysia is gathered in order to know how much the service actually cost. Further on, analysis is being made between the possible platforms and technologies that are to be used in developing the system.

3.2.1 SMS – Usage and Charges

SMS is very popular among the people in this country. Malaysians sent a stunning 21.03 billion *SMS* last year, more than twice as many as in 2004 [NST, 2006]. This suggests that Malaysians are becoming increasingly comfortable using this service because it is cheaper and easier than most other modes of communication. People opt for the service because they want to save cost and they feel more comfortable to send text messages rather than talking on the phone. The table below outlines the standard mobile call and *SMS* charges in Malaysia [Maxis, 2006].

Table 3.1: Call and SMS Rates in Malaysia

Calls	Peak Rate (7am to 7pm) (per minute)	Off-Peak Rate (7pm to 7am) (per minute)
Local Area	RM0.30	RM0.15
Outstation	RM0.30	RM0.30
SMS	RM0.15/sms	

3.2.2 Java 2 Micro Edition (J2ME)

The first question that comes to mind is why J2ME? Why not simply use the application designed in Java 2 Standard Edition (J2SE) or Java 2 Enterprise Edition (J2EE) for handheld devices? The answer lies in the basic design of these handheld devices. These devices are conceptually designed to be handy and compact; hence, they are small, lightweight, and portable. They have limited computing power, limited memory, a small display area, and limited input power (being battery operated). Besides, most of these handheld devices work on proprietary software, with little or no compatibility with other brands or other devices. Hence, what is required is a platform on which a memory-efficient, device-independent or platform-independent application can be built. An application designed in J2SE, for example, cannot run in a limited memory space of 16K–512K, which happens to be the typical range of memory for handheld devices. The solution lies in J2ME, the third platform (after J2EE and J2SE) offered by Sun Microsystems.

J2ME is Java's platform for embedded and small consumer electronic devices. The J2ME technology has been developed specifically to work within constrained resources (for instance, within a limited memory range of 128K–512K). It should, however, be noted that J2ME is not restricted to lower-end devices only. It can also be used on higher-end devices, such as set-top boxes, with as much computing power as a PC. Since J2ME is upwardly scalable to work with J2SE and J2EE, it enables these small consumer devices to be networked with servers or PCs.

The J2ME platform consists of a J2ME Virtual Machine and a set of APIs that are suitable for consumer and embedded devices. The J2ME technology can be divided into two primary components — configurations and profiles. These components can be understood if we think of J2ME in terms of a layered technology, with one layer working upon the other. The base layer is formed by a configuration, upon which operates the second layer, formed by a profile. Figure 2 illustrates this concept. A configuration is composed of the low-level APIs and the J2ME Virtual Machine; both of these provide an interface with a device's operating system. A profile built on top of a configuration is composed of APIs that provide functionality to build the user interface and develop the classes required to build an application.

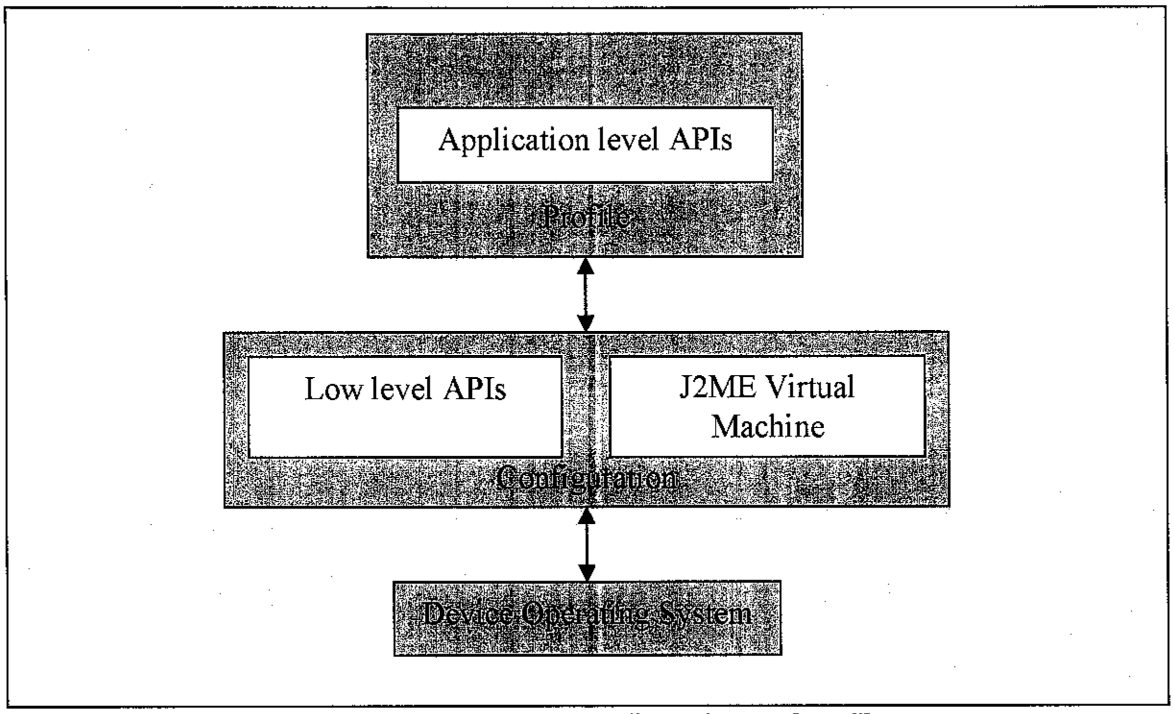


Figure 3.2: The concept of configurations and profiles

Think of a configuration as an abstract entity that provides basic J2ME functionality to a device, whereas a profile is what utilizes this configuration to allow the actual implementation of that functionality. For example, a configuration may support J2ME input/output functions on a family of devices, but the implementation of the input/output streams and their associated methods, properties, and so on depends upon the profile being used. Configurations and profiles are complementary to each other; both are required to develop and run a J2ME application.

J2ME Virtual Machine

As mentioned earlier, J2ME is used for devices with limited memory. This means that the Java 2 Virtual Machine (JVM) meant for PCs and servers cannot be used with low-end electronic devices such as mobile phones, two-way pagers, hand-held devices, screen phones, smart phones, and so on. In addition, J2ME targets high-end electronic devices such as set-top boxes, car navigation systems, and handheld PCs that have much better resources. However, they still don't accommodate the large size of the conventional JVM. Therefore, to support the J2ME technology, two smaller Virtual Machines have been developed. These are, the K Virtual Machine (KVM), which has a smaller footprint than CVM and is used with low-end devices, and the C Virtual Machine (CVM), which has a footprint larger than KVM and is used with high-end devices.

K Virtual Machine (KVM)

The K Virtual Machine (KVM) has been developed keeping in mind the constraints of small mobile devices being manufactured in the industry. The KVM is a highly optimized version of the conventional JVM, with a size as small as 50K. Since KVM was specifically designed for very small environments that are proprietary, it has also been made highly customizable to enable manufacturers to adapt it to suit their particular device. The design considerations for KVM ensure that it is capable of running on low-power processors. The KVM can run on any system that has a 16-bit/32-bit processor and 160-512 K of total memory. Nevertheless, the size reduction has occurred at the expense of a vast number of packages that are not supported by the KVM. As of now, the KVM has no support for certain features such as determinism, long and float data types, and so on.

The K Virtual Machine can theoretically run several profiles, but it cannot run perfectly all the profiles and APIs that aren't specifically designed for it, just as it cannot be used to run the Connected Device Configuration (CDC). It is meant for Connected Limited Device Configuration (CLDC). Presently, KVM supports only one profile — Mobile Information Device Profile (MIDP). This means that applications written for the more capable C Virtual Machine (CVM) or for the conventional JVM most probably cannot run on the KVM without some changes. However, the converse is not true — applications written for KVM can easily run on the CVM or the normal JVM.

J2ME configurations

J2ME configurations have been classified into two categories — Connected, Limited Device Configuration (CLDC) for low-end devices with 128K–512K memory and Connected Device Configuration (CDC) for 512K+ devices. For this project, CLDC is being used as mobile phones are categorized under the low-end devices group.

Connected, Limited Device Configuration (CLDC)

CLDC is meant for small devices such as mobile phones, with constrained resources. CLDC is ideally suited for devices with a 16/32-bit microprocessor and can work on an available memory as low as 160K. It uses the small K Virtual Machine (KVM) and a limited set of libraries. Together, the KVM and the libraries can be stored in just 128K of memory space. Limited functionality is the price paid for using the memory-efficient CLDC. For example, the most commonly used J2SE packages, such as java.lang.awt, java.lang.beans, and others, have been dropped. In fact, CLDC contains only the following four packages:

- `java.io`: A stripped-down version of the J2SE `java.io` package. It contains the classes required for data input and output using streams.
- `java.lang`: A stripped-down version of the J2SE `java.lang` package. It contains the classes that are basic to the Java language, such as the wrapper classes for data types.
- `java.util`: A stripped-down version of the J2SE `java.util` package. It contains classes such as `Calendar`, `Date`, `Vector`, and `Random`.
- `javax.microedition.io`: A newly introduced CLDC-specific class that defines the Generic Connection Framework. It contains the classes for handling all types of connections by using the same framework.

The emphasis in CLDC is providing just the basic functionality to conserve memory. Although certain basic features of J2SE are altogether missing in

CLDC, certain implementations have been altered to make them simpler. The following list discusses these features of CLDC:

- Data types `long` and `float` are not supported. All the methods of J2SE inherited classes that use these data types have been removed.
- The number of runtime errors has been reduced significantly for the classes included in CLDC. In fact, only the following three errors (`java.lang.Error`, `java.lang.OutOfMemoryError`, and `java.lang.VirtualMachineError`) are available. Other errors are handled in an implementation-specific manner.
- To make garbage collection simple, support for finalization is not provided. There is no finalizing method in the `java.lang.Object` class.
- Java Native Interface (JNI), which provides a means to access the local hardware, is not supported in CLDC. The purpose is to eliminate platform-dependence so that the applications can be ported to any platform containing the virtual machine.
- Threads can be used but not thread groups or daemon threads.
- In the standard edition, objects can be marked for possible garbage collection. This cannot be done with CLDC. In other words, there is no support for weak references.
- Verification of classes to check whether the code is well formed is done off-device — that is, on the desktop system on which the applications are developed — by a tool called *pre-verifier*. Pre-verification process should be done explicitly after compiling the code.
- A different security model is used in CLDC that is somewhat similar to the one used in browsers for downloaded applets. The reason is that the model used in the standard edition is too heavy for small devices, and the security needs of the connected devices are similar to those of the browsers.

MCS utilizes the capabilities of CLDC with Mobile Information Device Profile (MIDP) to provide the complete J2ME runtime environment for Java-enabled mobile phones.

3.2.3 Jabber

Jabber is best known as "the Linux of instant messaging" -- an open, secure, ad-free alternative to consumer *IM* services like AIM, ICQ and MSN. Under the hood, Jabber is a set of streaming XML protocols and technologies that enable any two entities on the Internet to exchange messages, *presence*, and other structured information in close to real time. Jabber technologies offer several key advantages:

- Open -- the Jabber protocols are free, open, public, and easily understandable; in addition, multiple implementations exist for clients, servers, components, and code libraries.
- Standard -- the Internet Engineering Task Force (IETF) has formalized the core XML streaming protocols as an approved *Instant Messaging* and *presence* technology under the name of XMPP, and the XMPP specifications have been published as RFC 3920 and RFC 3921.
- Proven -- the first Jabber technologies were developed by Jeremie Miller in 1998 [Jabber.org, 2006] and are now quite stable; hundreds of developers are working on Jabber technologies, there are tens of thousands of Jabber servers running on the Internet today, and millions of people use Jabber for *IM*.
- Decentralized -- the architecture of the Jabber network is similar to email; as a result, anyone can run their own Jabber server, enabling individuals and organizations to take control of their *IM* experience.
- Secure -- any Jabber server may be isolated from the public Jabber network (e.g., on a company intranet), and robust security using SASL and TLS has been built into the core XMPP specifications.
- Extensible -- using the power of XML namespaces, anyone can build custom functionality on top of the core protocols.

- Flexible -- Jabber applications beyond *IM* include network management, content syndication, collaboration tools, file sharing, gaming, and remote systems monitoring.
- Diverse -- a wide range of companies and open-source projects use the Jabber protocols to build and deploy real-time applications and services; you will never get "locked in" when you use Jabber technologies.

The Jabber server is far more complex than those of the conventional Client/Server architecture, whose simplicity may be attributed to the fact that they often overlook the client priorities. With the Jabber server, making Jabber clients for different platforms is far simpler and involves fewer headaches than with the conventional servers. In order to start creating a Jabber client, we need at first understand Jabber's architecture.

Jabber Architecture

Jabber's architecture is remarkably similar to the email architecture. Just like the email architecture, Jabber *Instant Messaging* works through a distributed network of servers using a standard protocol (SMTP in case of email, XMPP for Jabber). Anyone is free to run their own server with their own set of users. The user addresses are DNS bases and of the form user@host - again similar to the email. The distributed client/server architecture allows for a great degree of flexibility and interoperability. Users use a relatively simple client to connect to their server. Any messages that the user sends are sent to the server which relays them to other servers based on the address of the recipient. The messages are delivered immediately - opposed to the store-forward approach of email. Presence is a key feature of the system. When a user is connected to her server, the server knows she is online. This *presence* information is immediately available to other users who have subscribed for it provided that the user has authorized them to know this information.

Jabber has opted for client-server architecture as opposed to peer-to-peer architecture that is used by some alternative *Instant Messaging* systems. An important design guideline has been to move complexity to the server and keep the client as simple as possible. The client-server separation and putting the complexity in the server has several benefits:

- Clients do need to understand any other protocols if the recipient uses a different protocol.
- Clients need not be modified if the inter-server protocols are modified or new protocols are added.
- The contact lists and personal information is maintained at the server and a user can have access to this information with any client from any operating system. If this were kept at the client, a user using different clients will need to synchronize them routinely.

Client

The guiding design criterion for the Jabber client is simplicity. The only architectural requirements for the client are:

- It needs to understand the Jabber protocol to be able to speak to the Jabber server
- Support security mechanisms such as encryption for users that require secure communications
- Be easy to use as the end-user interaction only takes place with a client

Server

A consequence of keeping the client simple has been that all the important functionality resides in the server. It communicates with the clients and other Jabber servers using streaming XML protocols exchanging structured data and instant messages. It also allows for interoperability with other commercial *Instant Messaging* systems having proprietary protocols i.e. AIM, ICQ, IRC and WAP etc. The jabber server performs four key functions:

- Maintaining a list of registered users, their personal information and contact lists

- Listening for clients to open connections with the server, communicating with clients and relaying messages and *presence* information for them
- Communicating with other Jabber servers
- Translate between different protocols to interoperate with non-Jabber servers using proprietary protocols

Structure of Jabber XML protocol

As mentioned earlier, XML is the basis of the Jabber system. Communication between the client and the Jabber server takes place on port 5222. Two XML streams are involved in the exchange of data. One stream delivers the data packet from the client to the server, and the other stream delivers the data from the server to the client. The following code snippet is an example of XML exchange between the Jabber server and the client.

Listing 3.1: XML exchange between Jabber server and its client.

```
SEND:   to='aslan.net'
SEND:   xmlns='jabber:client'
SEND:   xmlns:stream='http://etherx.jabber.org/streams'>
RECV:  <stream:stream
RECV:   xmlns:stream='http://etherx.jabber.org/streams'
RECV:      id='38fd8070'
RECV:   xmlns='jabber:client'
RECV:   from='aslan.net'>
       (XML for user session goes here)
SEND:  </stream:stream>
RECV:  </stream:stream>
```

Jabber's Open XML protocol contains three top-level XML elements (also called tags).

1. *<presence/>* — This element determines the status of the user. The structure of the *presence* element is as follows:

Listing 3.2: Structure of the presence element.

```
<presence from='ayeen@aslan.net/JabberMCS' to='azlanf@aslan.net/JabberMCS'>
<status>Online</status>
</presence>
```

Based on the status of the user to be communicated, the *<presence>* element can be evaluated on the basis of the following values:

- probe — This value of the *presence* element is used to send a special request to the recipient of the message without waiting for the user's *presence* information. Notice that the server, not the client, processes such a request.
- subscribe — This sends a request that the recipient automatically send the *presence* information to the sender whenever the user changes its status.
- subscribed — This sends a notice that the sender accepts the recipient's request for *presence* subscription. The server now sends the recipient the sender's *presence* information whenever it changes.
- unsubscribe — If the value of the *presence* element is unsubscribed, the user sends the request to the recipient of the message to stop sending the messages of his/her *presence*.
- unsubscribed — In case the *presence* element holds this value, then it indicates that the user will not be able to communicate in any way with the sender of this message. In such a situation, the server no longer sends the sender's *presence* to the recipient of the message.

- from — This mentions the name or id of the sender of the message.
- to — This mentions the name of the recipient of the message.
- show — This displays the status of the user.
- status — This displays the description of the status.

2. *<message/>* — This element is used for sending the messages between two Jabber users. JSM (Jabber Session Manager) is responsible for catering all messages regardless of the status of the target user. If the user is online, the JSM will instantly deliver the message; otherwise, the JSM will store the message and deliver it to the user no sooner than he or she comes online. The <message> element contains the following information:

   - to — This identifies the receiver of the message.
   - from — This mentions the name or id of the message's sender.
   - text — This element contains the message about to be delivered to the target user.

Listing 3.3: Structure of the message element.

```
<message type='chat' from = 'ayeen@aslan.net/JabberMCS'
to ='aslan@aslan.net/JabberMCS'>
<body>Hello!</body>
</message>
```

3. *<iq/>* element — This element manages the conversation between any two users on the Jabber server and allows them to pass XML-formatted queries and respond accordingly.

   The main attribute of the *<iq/>* element is type. The type attribute of the <iq/> element can carry the following values:

   a. get — This attribute of the *<iq/>* element is used to retrieve the values of the fields present.
   b. set — This attribute is responsible for setting or replacing the values of the fields queried by the get attribute.
   c. result — This attribute indicates the successful response to an earlier set/get type query.

Listing 3.4: Structure of the iq element.

```
<stream:stream id='38fd8070' from='aslan.net' xml:lang='en' xmlns='jabber:client'
xmlns:stream='http://etherx.jabber.org/streams'>
<iq to='ayeen@aslan.net/JabberMCS' id='s3' type='result'><query xmlns='jabber:iq:roster'>
<item name='aslan' subscription='both'
jid='aslan@aslan.net'><group>love</group></item><item name='mfuzze' subscription='both'
jid='mfuzze@aslan.net'><group>love</group></item></query></iq>
</stream:stream>
```

3.3 Design Phase

MCS is an instant messenger application that uses Jabber technology as its standard and protocol. The whole purpose of *MCS* development is to provide an alternative of text messaging for Java-enabled mobile phone users. It is based on J2ME (MIDP 2.0) and the MicroJabber library [MicroJabber, 2005]. *MCS* is being designed to run on all Java-enabled mobile phones with the abilities to support all Jabber client basic features and to provide easy, fast and essential GUI.

MCS is built from different Java packages which each have their own classes to handle specific tasks. The application is consists of these packages:

`mcs`: Contains the GUI Midlet (main program) and the Jabber stanzas reader
`mcs.connection`: Contains the connection tools
`jabber.roster`: Contains classes concerning Jabber ID and roster management
`jabber.conversation`: Contains classes to manage chats
`jabber.presence`: Contains classes for jabber presence management
`jabber.subscription`: Contains classes for registration to jabber servers and Jabber IDs subscriptions
`org.bouncycastle.crypto.digests`: Contains classes for password encryption
`util`: Contains utility classes
`xmlstreamparser`: Contains classes for XML parsing

The communication of these packages is explained in Figure 3.3.

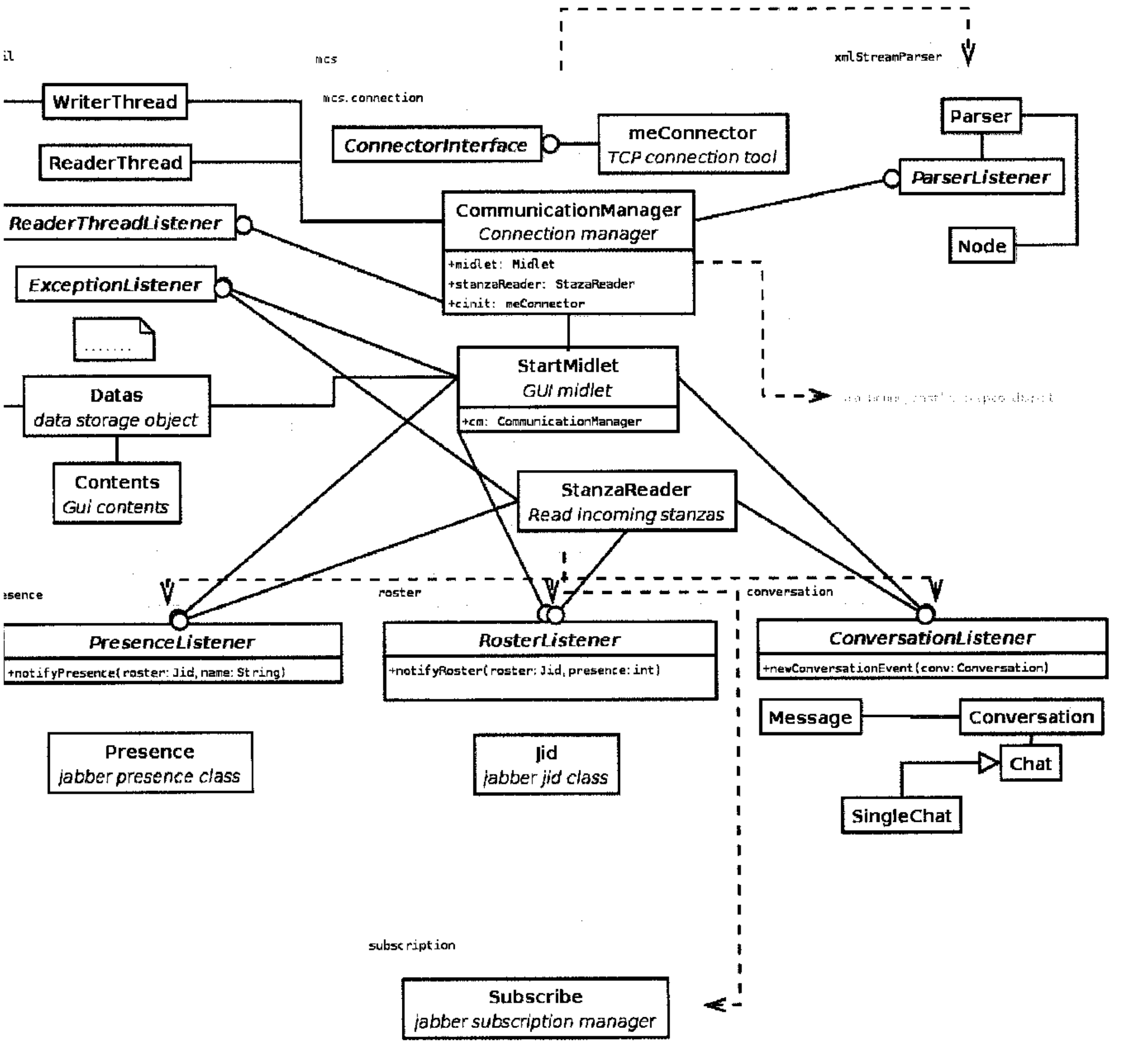


Figure 3.3: Communication between MCS Java Packages

3.3.1 Programming MCS

In *MCS*, two J2ME classes work together to recognize user events and to send user requests to the Jabber server: `mcs.StartMidlet` and `mcs.StanzaReader`. The midlet `mcs.StartMidlet` provides the user interface necessary for the working of *MCS* application, and the `mcs.StanzaReader` class works in the background, reading incoming stanzas and answers them if necessary (as method result). Stanza in this context refers to first depth XML nodes. *MCS* uses XML requests that comply with the XMPP protocol to communicate with the Jabber server. This enables *MCS* to connect with any Jabber server which then allows *MCS* to accept contacts from other servers running the same protocol. Some of the servers are jabber.org, ijabber.com and gmail.com. Full list of public Jabber servers is included in the appendix.

1. The `mcs.StartMidlet` GUI is based on the `ChoiceGroup` class which is much the same as radio buttons in Java Swing applications. To create the GUI, methods are called and command listener is added to the list. The code snippet below shows how this is being done. It provides an “OK” and “Exit” button.

Listing 3.5: Creating the GUI of Offline Menu

```
public Displayable getGuiOfflineMenu() {
    Form res = Contents.offline_form;
    if (res.size() > 0)
    return res;

    offLineMenu = new ChoiceGroup("OffLine",  List.EXCLUSIVE, Contents.offlineChoices, null);

    res.append(offLineMenu);
    res.addCommand(Contents.ok);
    res.addCommand(Contents.exit);
    res.setCommandListener(this);
    return res;
}
```

2. In the above example, four parameter values have been passed while creating a `ChoiceGroup` object. The first value containing the string “Offline” is the name of the list. The second value, `List.EXCLUSIVE` denotes the type of list. In this case, the list type only allows one option to be selected. The third value, `Contents.offlineChoices`, is a string array containing values for the list elements (here these elements are "Connect", "Login Info" and "Help"). The fourth value that is a null parameter here accepts an image array for any image icons. The offline display on the screen is as shown in Figure 3.5.

3. When the user presses the "OK" button, the corresponding screen of the list item in focus appears. The index of the selected item is obtained to determine the list element that the user has focused upon before pressing "OK". The "Connect" option establishes connection to the Jabber server, "Login Info" option displays login details screen to get input from user and the "Help" option displays the help screen.

Listing 3.6: Command Listener for the Offline Menu

```
public void commandActionOfflineMenu(String id) {
    if (id.equals("ok")) {
            display.setCurrent(getGuiWaitConnect());
            internal_state = WAIT_CONNECT;

            if (offLineMenu.getSelectedIndex() == 0)
                    cm.connect(0); //user already registered
            else if (offLineMenu.getSelectedIndex() == 1) {
                    display.setCurrent(getGuiParams());
                    internal_state = PARAMS;
            }
            else if (offLineMenu.getSelectedIndex() == 2) {
                    display.setCurrent(Contents.help,getGuiOfflineMenu());
                    internal_state = OFFLINE;
            }
            else
                    System.out.println("Error: choice not chosen");
    }
    else if (id.equals("exit")) {
            notifyDestroyed();
    }
}
```

4. The GUIs for index 1 and 2 in the code snippet above are shown in Figure 3.6 and Figure 3.7. In the "Login Info" screen, if the user is connecting to a public Jabber server such as ijabber.com, there is (in general) no formal process for requesting a username. All the user has to do is try to log in with his/her desired Jabber ID and password. If it doesn't work (e.g., because the Jabber ID is taken), he/she has to try again with other Jabber ID. (If the user is using a private Jabber server, for example a server running on a company's intranet, he/she may need to contact the server administrator in order to register an account.)

5. After the user selects the “Connect” option, connection to the Jabber server is established. Communication between *MCS* and the Jabber server takes place on port 5222. The connection is handled by the `mcs.connection.meConnector` class which extends `mcs.connection.ConnectorInterface` thread class. The `mcs.connection.meConnector` class receives parameters from the `mcs.connection.CommunicationManager` class (Listing 3.7). The code snippet below shows how the parameters are passed.

Listing 3.7: Passing parameters to establish connection

```
public void connect(int state) {
        cinit = new meConnector(“aslant.net”, 5222, this);
        type_of_connection = state;
        cinit.start();
}
```

The argument `state` takes the value 0 of type `int` from `mcs.StartMidlet` class as indication that user has already been registered. Three parameter values have been passed while creating a `mcs.connection.meConnector` object. The first value indicates the hostname, the second value states which communication port to be used and the third value represents the `mcs.connection.CommunicationManager` object. The start() method of `mcs.connection.meConnector` class initiates the connection with the Jabber server. The start() method is referred to as run() method in the `mcs.connection.meConnector` class (Listing 3.8).

Listing 3.8: Calling the method run to initiate connection with Jabber server

```
public void run() {
        StringBuffer connectorStringBuffer = new StringBuffer( "socket://");
        connectorStringBuffer.append( hostname );
        connectorStringBuffer.append( ":" );
        connectorStringBuffer.append( _port );
        connectorStringBuffer.append( "" );
        String connectorString = connectorStringBuffer.toString();
        System.out.println(connectorString);
        try {
            connection = (StreamConnection) Connector.open( connectorString );
            _cm.notifyConnect(connection, this.openInputStream(), this.openOutputStream());
        }
        catch (Exception e){e.printStackTrace();
            System.out.println("Connection Error:"+ e.getMessage());
            _cm.notifyNoConnectionOn("Connection Error:"+ e.getMessage());
        }
        return;
    }
```

6. Upon successful connection, the user's contact list is displayed on the mobile phone's display, as shown in Figure 3.8. The *presence* information of each of the user's contacts is represented by an image next to the contacts' Jabber ID.

   The list of friends shows the names of the friends along with their respective status. The friend list is also derived from the `ChoiceGroup` class and is of `EXCLUSIVE` type. The Options button in the left hand corner provides options such as disconnecting from server, adding contact to roster and changing status as shown in Figure 3.9.

   The user can now choose any option from the list. And the display on the mobile phone will show the corresponding GUI. While the Select button in the right hand corner provides options like deleting contact and sending message. This is shown in Figure 3.10. The index of the selected friend is obtained to associate the chosen option with the particular friend. For example, if the contact *mfuzze@aslan.net* is selected and the *Send Message* option is chosen, the program takes care that the message keyed in is subsequently sent to *mfuzze@aslan.net*.

7. When a user wants to start a conversation with a contact, a message window like the one shown in Figure 3.11 appears. The user can type his/her message in the text box and press "Send" to send it.

8. When a user receives a message from a friend, a notification like Figure 3.12 is shown. This allows the user to be notified by the program whenever other contacts send messages to the user. Notifications are also shown when users in the contact list are "online", "away", or "dnd".

9. When a user chooses the option "Add Contact" as in Figure 3.8, a form like Figure 3.13 is shown to receive input from the user. The user has to insert his/her friend's Jabber ID and assign which group that the friend belongs to. After the user had entered all the details and presses the "OK" button, an add-friend request is generated by the `jabber.subscription.Subscribe` class and is send to the Jabber server.

   When other user tries to add the user as a friend, a form like the in Figure 3.14 appears that provide the option of accepting ("Accept" button) or declining ("Deny" button) the other user as a friend.

10. A user may also change his/her current status by selecting the option "Change Status" in Figure 3.8. The status option list is shown in Figure 3.15. This provides the user's *presence* information to his/her contacts.

All requests are handled by the `mcs.StartMidlet` class that in turn calls the appropriate classes and methods to generate the appropriate XML streams. These requests are then being sent to the Jabber server for responses. The full workings of every J2ME classes used in *MCS* are explained in the Java Documentation included with this report.

3.4 Implementation Phase

Implementation which is the final phase of the project work involves execution and testing of *MCS* on the emulator to ensure it performs as designed. After the process has been successful, a user manual for the application is written to guide users on how to use the application. Finally, *MCS* is being installed into a Java-enabled mobile phone as the final process of the implementation phase.

3.4.1 Execution and Testing

The execution and testing of *MCS* have been done using the J2ME Wireless Toolkit Emulator. The emulator simulates an MIDP device on the desktop computer. It is a convenient way to see how the application performs in an MIDP environment. Below are the screenshots of the *MCS* application when it is run using the emulator. The functions of each screenshots are the same as explained in the Design section.

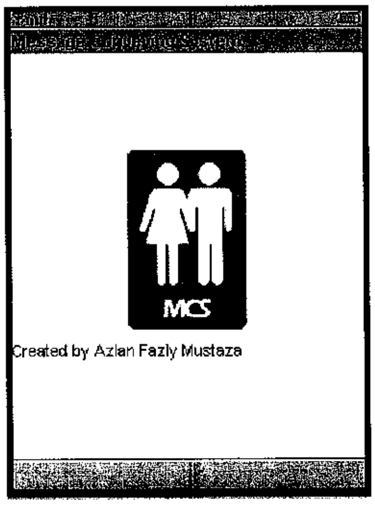


Figure 3.4: MCS splash screen

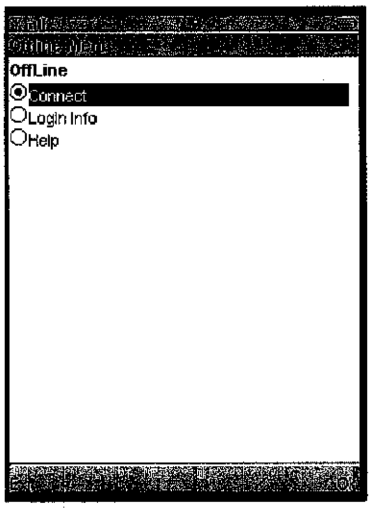
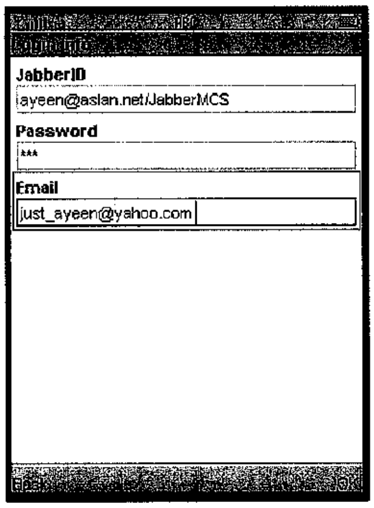


Figure 3.5: MCS offline screen



JabberID

ayeen@aslan.net/JabberMCS

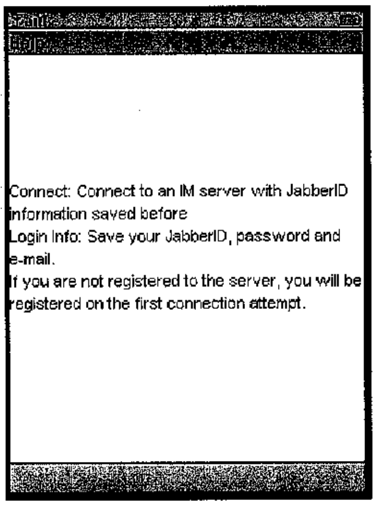
Password

***

Email

just_ayeen@yahoo.com

Figure 3.6: GUI for "Login Info" option.



Connect: Connect to an IM server with JabberID information saved before
Login Info: Save your JabberID, password and e-mail.
If you are not registered to the server, you will be registered on the first connection attempt.

Figure 3.7: GUI for "Help" option.

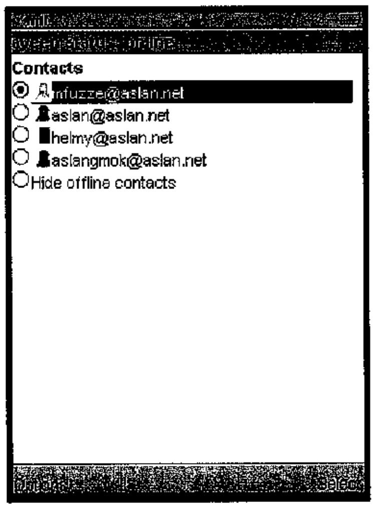


Figure 3.8: MCS main screen

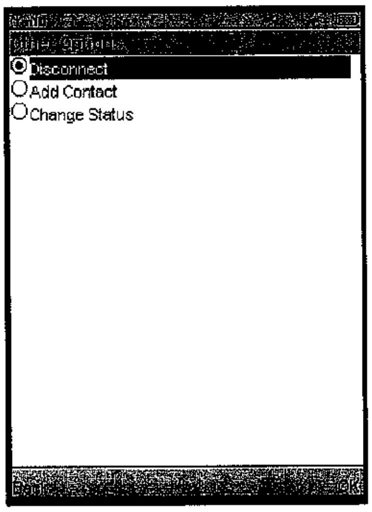


Figure 3.9: Online Options list

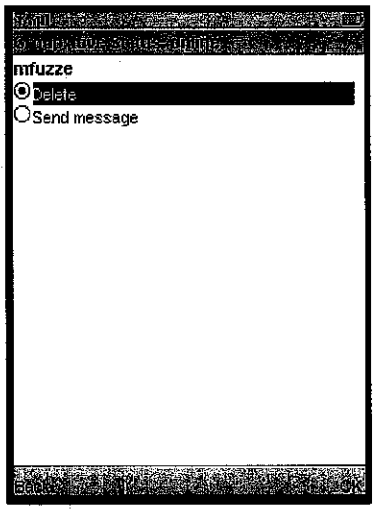


Figure 3.10: Contact Options list

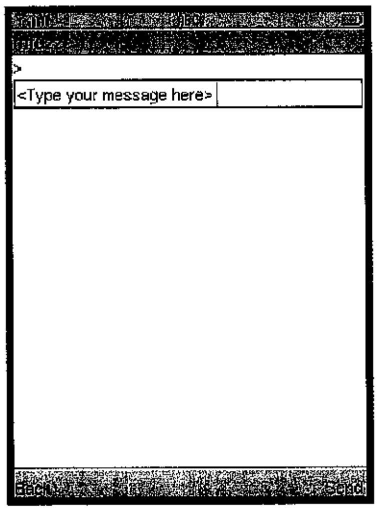


Figure 3.11: The GUI to send a message

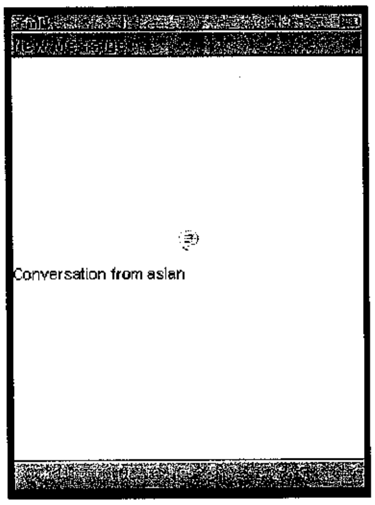
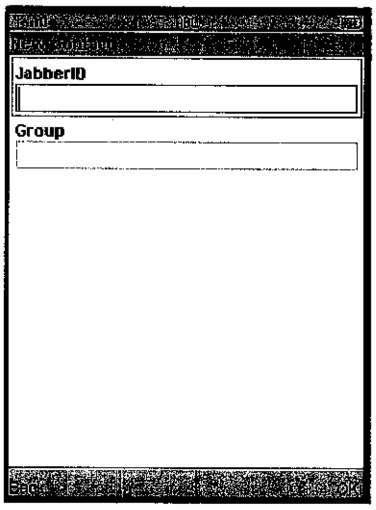


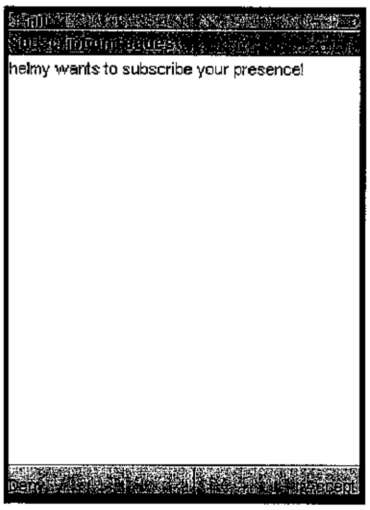
Figure 3.12: Notification message of incoming conversation



JabberID

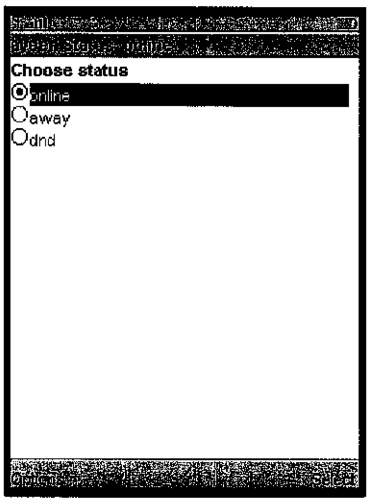
Group

Figure 3.13: The Add Contact Form



helmy wants to subscribe your presence!

Figure 3.14: The Subscription Request Form



Choose status

[✓ online]
away
dnd

Figure 3.15: The Status Option List

3.4.2 MCS User Manual

After the execution and testing had been successful, the user manual of *MCS* is written. The full user manual is available as **Appendix A**.

3.4.3 Installation

MCS runs on all mobile phones which support J2ME with MIDP 2.0 and CLDC 1.1. The device should also support GPRS for the application to connect to the Jabber server. The installation of *MCS* is very easy as you only have to deploy the .jar file in your device following the specific guidelines provided by the device manufacturer. For demo purpose, I have used a Motorola SLVR unit as the target device. The following diagram shows the device installing the MCS.jar file.

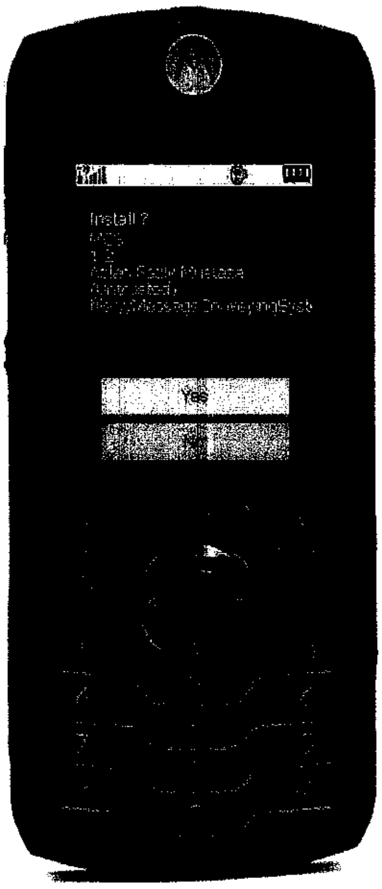


Figure 3.16: MCS installation on Motorola SLVR

After the installation has been completed, *MCS* can be run and users can connect to any Jabber server that is available on the internet. The full list of the public Jabber server is included as **Appendix B**.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 MCS VS SMS

Over the years, *Instant Messaging* has proven itself to be a feasible technology not only to fun-loving people but to the world of commerce and trade, where quick responses to messages are crucial. With the wide acceptance it commands, *Instant Messaging* has created a major league of fans that uses *IM* as an ideal tool for day-to-day communication.

The step to bring *IM* into mobile phones has created a new way of text messaging to the people. Apart from being an alternative to calls and *SMS*, *MCS* allows spontaneous interaction among users over the network, which *SMS* lacks off. It also makes it easier in accessing remotely located users. Users from all around the world can use *MCS* to communicate with each other as long as they are within the coverage area.

With *MCS*, one can be informed about the *presence* of their contacts through the status information. This *presence* information does not exist on cell phones, but the minute they are adopted there, they will enable real-time communication at another level. People will be able to see if the person they want to reach is free to speak, and to decide if they're rather send him a written message. This gives *MCS* an advantage over *SMS* because *SMS* does not provide the users' *presence* information to be available to others. Table 4.1 shows the *presence* information provided by *MCS*.

Table 4.1: MCS Presence Information

Status	Description
👤	Online
👤	Away
▮	Do Not Disturb
👤	Offline

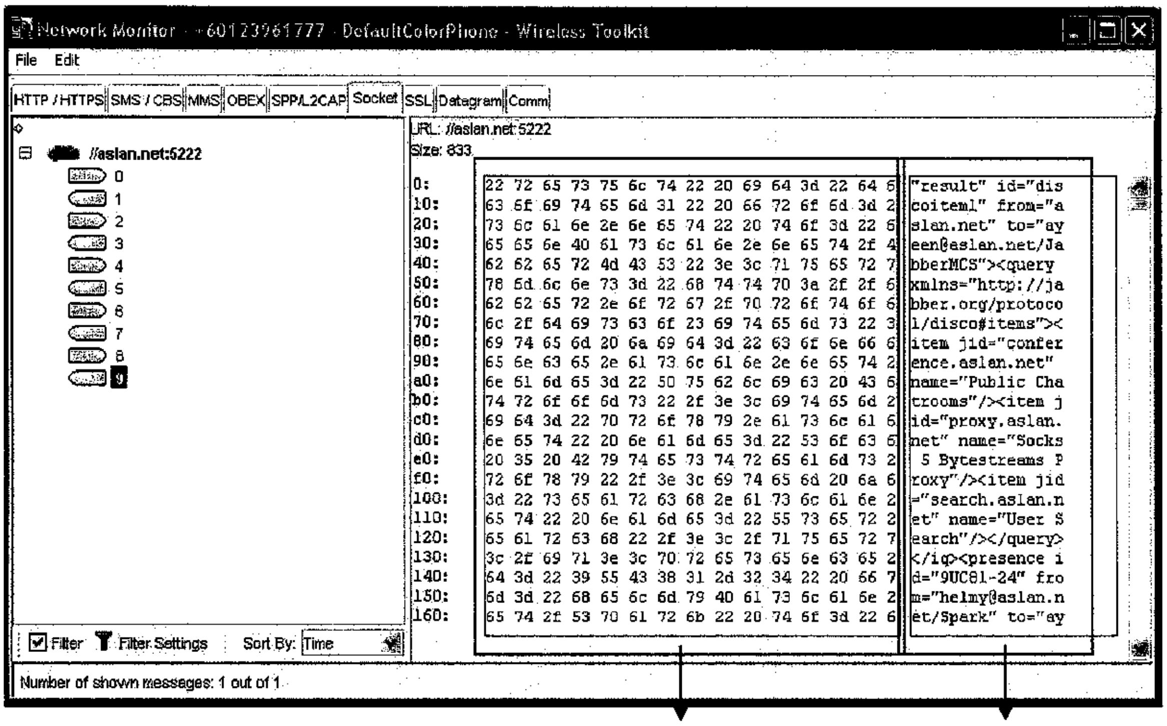
4.2 MCS as a cheaper text messaging communication

MCS establishes its connection to the Jabber server through General Packet Radio Service (GPRS) that is a connectivity solution based on Internet Protocols that supports a wide range of enterprise and consumer applications. With throughput rates of up to 40 kilobit per second, users have a similar access speed to a dial-up modem, but with the convenience of being able to connect from anywhere. With GPRS, users can always be online. No more wasting time with dialing up, getting busy signals or disconnected. The connection is always on, always instantly available the moment they need it.

GPRS charges are based on packet counting which means a user is being charged based on the amount of data transferred. Thus with GPRS, a user only pays for the actual data sent and received, which is calculated in kilobytes (kB).

In Malaysia, the average cost for mobile phone calls is RM 0.30 per minute and RM 0.15 per message for *SMS* which can take up to 160 characters. As for GPRS, users are being charged RM 0.10 for each 10 kB of data transferred.

MCS sends requests and receives responses from the Jabber Server in XML format. XML encodes characters using UTF-8 encoding standards [Tbray.org, 2003]. In UTF-8, characters whose value is less than 128 (i.e. ASCII) are encoded as themselves in one byte. Thus, every character in the XML streams is considered as one byte. The figure below shows the XML streams involved and its characters representation in UTF-8 encoding.



```
URL: //aslan.net:5222
Size: 833

0:    22 72 65 73 75 6c 74 22 20 69 64 3d 22 64 6  "result" id="dis
10:   63 6f 69 74 65 6d 31 22 20 66 72 6f 6d 3d 2  coiteml" from="a
20:   73 6c 61 6e 2e 6e 65 74 22 20 74 6f 3d 22 6  slan.net" to="ay
30:   65 65 6e 40 61 73 6c 61 6e 2e 6e 65 74 2f 4  een@aslan.net/Ja
40:   62 62 65 72 4d 43 53 22 3e 3c 71 75 65 72 7  bberMCS"><query
50:   78 6d 6c 6e 73 3d 22 68 74 74 70 3a 2f 2f 6  xmlns="http://ja
60:   62 62 65 72 2e 6f 72 67 2f 70 72 6f 74 6f 6  bber.org/protoco
70:   6c 2f 64 69 73 63 6f 23 69 74 65 6d 73 22 3  l/disco#items"><
80:   69 74 65 6d 20 6a 69 64 3d 22 63 6f 6e 66 6  item jid="confer
90:   65 6e 63 65 2e 61 73 6c 61 6e 2e 6e 65 74 2  ence.aslan.net"
a0:   6e 61 6d 65 3d 22 50 75 62 6c 69 63 20 43 6  name="Public Cha
b0:   74 72 6f 6f 6d 73 22 2f 3e 3c 69 74 65 6d 2  trooms"/><item j
c0:   69 64 3d 22 70 72 6f 78 79 2e 61 73 6c 61 6  id="proxy.aslan.
d0:   6e 65 74 22 20 6e 61 6d 65 3d 22 53 6f 63 6  net" name="Socks
e0:   20 35 20 42 79 74 65 73 74 72 65 61 6d 73 2   5 Bytestreams P
f0:   72 6f 78 79 22 2f 3e 3c 69 74 65 6d 20 6a 6  roxy"/><item jid
100:  3d 22 73 65 61 72 63 68 2e 61 73 6c 61 6e 2  ="search.aslan.n
110:  65 74 22 20 6e 61 6d 65 3d 22 55 73 65 72 2  et" name="User S
120:  65 61 72 63 68 22 2f 3e 3c 2f 71 75 65 72 7  earch"/></query>
130:  3c 2f 69 71 3e 3c 70 72 65 73 65 6e 63 65 2  </iq><presence i
140:  64 3d 22 39 55 43 38 31 2d 32 34 22 20 66 7  d="9UC81-24" fro
150:  6d 3d 22 68 65 6c 6d 79 40 61 73 6c 61 6e 2  m="helmy@aslan.n
160:  65 74 2f 53 70 61 72 6b 22 20 74 6f 3d 22 6  et/Spark" to="ay
```

Number of shown messages: 1 out of 1

Figure 4.1: MCS XML Streams and UTF-8 Encoding

To explain further, I will give an example of the calculation for the number of bytes needed for a connection from the *MCS* application to the Jabber server and the usage charges for the connection.

Establishing a connection to the Jabber server:		
No. of characters in XML streams	=	2431 characters
Encode using UTF-8 (one byte for one character)	=	2431 bytes
Convert to kilobytes	=	2.431 kilobytes
Usage charges of GPRS	=	2.421/10 X RM 0.10
	=	RM 0.02

In the result acquired from above, the usage charges for the connection is RM 0.02 which is way cheaper than the average cost of a single *SMS* of RM 0.15. For other XML streams such as sending messages, adding contacts, changing status and so on, the number of characters in the XML streams is much less than establishing a connection. Therefore, I can honestly say that the total amount for all the streams will not be more than RM 0.15. This proves that *MCS* is a much cheaper text messaging communication as we are not paying for the service but paying for the amount of data transferred. Users can send messages to their contacts much cheaper than the conventional *SMS*.

4.3 Interoperability among different Jabber clients

One of the objectives of the development of *MCS* is to accept communication from different *IM* application that uses the same protocol. It is being achieved by using an established *Instant Messaging* protocol that uses open and XML-based protocols that provides the standard functionality of an *IM* system. This is where the Jabber XMPP protocol comes into the picture. As explained in Chapter 3, Jabber protocol allows different Jabber clients to communicate with each other through its distributed client/server architecture. It means that users who are using Jabber clients other than *MCS* can "talk" and see the *presence* information of an *MCS* user. Among the Jabber clients available are GoogleTalk, Meebo, Spark, Exodus etc. They each connect to different Jabber servers but still able to communicate with each other because they comply with the Jabber protocol. The same goes for *MCS* where it supports the capabilities of "meeting" with others no matter what Jabber client other users use.

CHAPTER 5
CONCLUSION

Presence and mobile *Instant Messaging* are clearly very promising and have already attracted a great interest. The establishment of GPRS technology will facilitate persistent 'always-on' connection with friends, family and colleagues through mobile *IM*.

This report has explained the concept and development of *MCS* as a mobile *IM* application system and the benefits in regards to mobile *IM* and *presence*. *MCS* has created a text messaging alternative using mobile phones. It also had achieved all the project's objectives stated in the early part of this report. It provides capabilities such as *presence* information that *SMS* lacks off and interoperability between different *IM* services that use the Jabber protocol. With such capabilities, *MCS* may become the new main stream of text messaging. The reasons for this are the wide acceptance of *IM* among the people nowadays and the low usage charges that it provides which is way cheaper than the conventional *SMS*. I hope that in the near future, people will opt for *MCS* rather than *SMS* to deliver text messaging to their families and friends. I believe that *MCS* will create its own major league of fans that uses *MCS* as an ideal tool for day-to-day communication.

REFERENCES

American.edu (2004). *Information Technology Landscape of Malaysia.* Accessed from the web page with URL: <http://www.american.edu/initeb/ym6974a/telecom.htm#Quick%20Links>

Antypas, J. & Leung, K. (2005). *Primer for Mobile Commerce: The Decision Making Process.*

Chakraborty, R. (2002). *Presence: A Disruptive Technology,* JabberConf 2001 presentation, Denvor.

Ericsson, Motorola, Nokia (2002). *Wireless Village: The Mobile IMPS Initiative.* Accessed from the web page with URL: <www.wireless-village.org>

FunMail (2002). *FunMail: Content Driven Wireless Messaging Solutions.* Accessed from the web page with URL: <http://www.funmail.com/>

Gsmworld.com (2006). *GPRS Platform.* Accessed from the web page with URL: <http://www.gsmworld.com/technology/gprs/index.shtml>

Haggar, P. (2000). *Practical Java Programming Language Guide.* Addison-Wesley.

IETF (2002). *Overview of the IETF.* Accessed from the web page with URL: <http://www.ietf.org/overview.html>

IETF (2001). *SIP: Session Initiation Protocol.* The Internet Society. Accessed from the web page with URL: <http://www.cs.columbia.edu/sip/drafts/draft-ietf-sip-rfc2543bis-03.pdf>

IMPP (2002). *Instant Messaging and Presence Protocol (IMPP).* IETF. Accessed from the web page with URL: <www.ietf.org/html.charters/impp-charter.html>

Jabber.org (2006). *Jabber People.* Accessed from the web page with URL: <http://www.jabber.org/people/jer.shtml>

Maxis (2006). *Maxis Call Charges in Malaysia.* Accessed from the web page with URL: <http://www.maxis.com.my/personal/mobile/call_charges/planscharges.asp>

MicroJabber (2005). *MicroJabber Project.* Accessed from the web page with URL: <http://micro-jabber.sourceforge.net/?microJabber>

Nardi, B., Whittaker, S., et al. (2000). *Interaction and Outeraction: Instant Messaging in Action.* CSCW'2000, ACM Press.

NST (2006). *The 21 Billion SMS Phenomenon*, Kuala Lumpur, New Straits Time.

PAM (2002). *About the PAM Forum.* PAM Forum. Accessed from the web page with URL: <www.pamforum.org/SubNav/background.html>

PAM (2002). *PAM Specification Document: version 1.0.* PAM Forum. Accessed from the web page with URL: <www.pamforum.org>

Peretz, M. (2002). *IM 101: Different Wireless Messaging Flavors.* Instant Messaging Planet. Accessed from the web page with URL: <www.instantmessagingplanet.com/wireless>

Pulver.com (2001). *The marriage of SMS and Instant Messaging.* Accessed from the web page with URL: <http://pulver.com/reports/smsim.html>

Tbray.org (2003). *Characters VS Bytes.* Accessed from the web page with URL: <http://www.tbray.org/ongoing/When/200x/2003/04/26/UTF>

Tulloch, J. (2002). *Communication Breakdown.* Mobile Communications International, Telecoms.com. 01.

Ulfelder, S. (2001). *Showdown over Instant Messaging.* M-Business, 68.

Wiral (2001). *Wireless Instant Messaging: Market Opportunity and Business Models.*

Woods, B. (2002). *Interoperability: Big Challenge for Mobile Messaging.* Accessed from the web page with URL: <http://www.instantmessagingplanet.com/wireless/ article.php/964591>

XMPP. (2002). *Extensible Messaging and Presence Protocol (XMPP).* IETF. Accessed from the web page with URL: <http://www.ietf.org/html.charters/xmpp-charter.html>

Yuan, M.J. (2003). *Enterprise J2ME: Developing Mobile Java Applications.* Prentice Hall.

Appendix A
MCS User Manual

Message Conveying System

User Manual

Author: Azlan Fazly Mustaza

Introduction

Message Conveying System (MCS) is an Instant Messaging client for Jabber protocol for GPRS mobile phones. It is based on J2ME (MIDP 2.0) and the MicroJabber library. According to the still quite low performances of smartphones, MCS has these targets: to support all the Jabber clients' basic features; an easy, fast and essential GUI; easy extendibility so that other features will be able to be added when more powerful devices will be on the market.

System Requirements: MCS runs on all GPRS mobile phones which support J2ME with MIDP 2.0 and CLDC 1.1.
Installation: MCS is very easy to install, you have to deploy the .jar file in your device following the specific guidelines provided by the device manufacturer.

Offline menu

When you start the application, after the preliminary splash screen, MCS shows the Offline menu.

- Connect
- Login Info
- Help

If you have never used MCS before, you have to set your connection options: choose "Login Info" and press OK, a form is shown where you can write your Jabber ID (JID), password, and email address (optional); these settings are stored in your device memory, so it's not necessary to write them each time you use MCS. Now you are ready to connect, select the Connect option!

If you are not already registered to the Jabber server, MCS tries to register your JabberID, if the registration is successful, the "Online" menu is displayed, otherwise an alert is shown and you must change your settings.

If you choose Help and press OK, a help alert is displayed for ten seconds.
On the left part of the screen there is an EXIT button for you to close the application.

Online screen

This screen shows your Roster (your contacts list), if you have any. In particular, only online contacts' JabberIDs are displayed. If you want to see the entire list you can choose "Show all Contacts" and press "Select".
In the Roster list each contact's JabberID has on its left an image which represents the user current status. For more details see the table below:

Status	Description
●	Online
○	Away
▮	Do Not Disturb
●	Offline

On the bottom left of the screen, there is the "Options" command. If you press this button, the Other Options menu is shown.

Other Options menu

This menu is composed by the following items:

- Disconnect: Disconnects from the server and return to the Offline menu.
- Add contact: This item allows you to add a new contact to your roster; a form is displayed to enter the JabberID, if the JabberID exists in the server, the new contact is added to the roster.
- Change Status: This item allows you to manage your present state, when this choice is selected MCS shows a menu with all the states you can choose: "online", "away" and "dnd".

Contact details

This screen contains the possible interactions that you can activate with the contact selected. On the top of the screen MCS shows the current state of the contact. These are the options for the selected contact.

• Delete: if you want to delete the contact.
• Send a message: send an instant message to the contact, MCS opens the Conversation screen (see Conversation screen section).

This last item can be replaced by "Active Conversation" if there is an active chat with this contact. If you select this item, you will enter the Conversation screen (see Conversation screen section).

Conversation screen

There is a form where you can insert the text to be sent (press the "Send" button on the right). Under the textbox is the history text of the conversation growing as a stack (last message on the top). On the left there is the "Back" button to return to the Online menu.

When there is an incoming message while MCS is showing a screen different from the Conversation screen, an alert will be displayed.

MCS supports six types of smileys. The smileys are displayed if you or the other user types the character representation of the smileys during a conversation.

Smileys	Character Representation	Description
☺	“:)” or “:-)”	Smile
☹	“:(” or “:-(”	Sad
😃	“:D” or “:-D”	Big Smile
😛	“:p” or “:-p”	Tongue
😉	“;)” or “;-)”	Wink
😮	“:0” or “:-0”	Shock

Appendix B

Jabber Public Servers

This is the list of the Jabber/XMPP servers that are registered with the XMPP Federation. You can use MCS to connect to any of these servers.

* Source: http://www.jabber.org/user/publicservers.php

Country	Domain	Latitude	Longitude
at	jabber.uplink.at	48.14	16.23
at	jaim.at	48.12	16.22
at	jabber.linuxlovers.at	46.65	14.35
at	selfnet.at	48.12	30.01
au	jabber.org.au	-31.97	115.82
au	jabber.oztralia.com	-33.50	151.10
au	ekto.ath.cx	-27.29	153.80
be	jabberweb.be	50.88	4.70
bg	jabber.networx-bg.com	43.20	24.30
bg	jabber.minus273.org	43.12	25.41
br	redesul.net	-25.25	-49.10
br	chat.placeredes.com.br	-23.33	-46.39
ca	dhbit.ca	49.13	-123.21
ca	jabberquebec.net	46.03	-73.46
ca	pan.ubishops.ca	45.22	-71.51
ca	jabbercanada.net	46.03	-73.46
ca	globalrelay.net	49.21	-123.10
ch	swissjabber.ch	46.57	7.28
ch	swissjabber.li	46.57	7.28
ch	swissjabber.de	46.57	7.28
ch	swissjabber.org	46.57	7.28
cz	jab.undernet.cz	50.22	12.87
cz	jabber.cz	50.06	14.26
cz	njs.netlab.cz	50.06	14.26
cz	jabbim.cz	50.06	14.26
cz	jabbim.sk	50.06	14.26
cz	jabbim.com	50.06	14.26
de	deshalbfrei.org	49.27	11.05
de	jabber.i-pobox.net	52.24	9.44
de	jabber.ccc.de	48.38	10.00
de	aszlig.net	49.00	8.23
de	jabber.schwarzvogel.de	51.13	6.47
de	jabjab.de	49.27	11.04
de	amessage.info	49.23	11.11

de	amessage.at	49.23	11.11
de	amessage.be	49.23	11.11
de	amessage.ch	49.23	11.11
de	amessage.de	49.23	11.11
de	amessage.dk	49.23	11.11
de	amessage.li	49.23	11.11
de	amessage.nl	49.23	11.11
de	jabber.hot-chilli.net	52.30	13.17
de	foxalpha.de	53.33	9.58
de	jabber.23-42.net	49.27	11.03
de	jabber.sow.as	52.24	9.44
de	jabber.fourecks.de	52.24	9.44
de	jabber4friends.de	49.27	11.05
de	jabber.workaround.org	53.60	10.10
de	jabber.cc	49.27	11.03
de	camaya.net	49.27	11.03
de	jabber.freenet.de	51.14	6.42
de	draugr.de	49.27	11.50
de	im.flosoft.biz	52.30	13.25
de	2on.net	49.00	8.23
de	jabber.servequake.com	49.00	8.23
de	phcn.de	48.11	11.45
de	mydebian.de	50.12	8.73
de	schokokeks.org	51.28	7.13
de	alpha-labs.net	51.52	7.45
de	funkyjoh.de	49.85	7.87
de	jabber.pilgerer.de	48.13	11.58
de	nerdtreff.de	49.45	11.07
de	hellercomserver.dyndns.org	52.30	13.25
dk	jabber.dk	55.40	12.34
es	jabber-hispano.org	42.21	-8.75
es	im.rediris.es	40.44	-3.68
es	tecnisol.com	37.45	-6.17
es	udias.com	43.27	3.48
es	xintcat.org	42.22	1.28
es	jabberland.com	42.61	-5.60
es	jabbernet.es	39.52	-3.00
fr	jabber.netrusk.net	43.60	1.40
fr	jabber.tcweb.org	50.63	3.06
fr	mithrandir.net	48.48	2.20
fr	m0g.net	48.48	2.20
fr	pratz.net	48.48	2.20
fr	kaya.epiknet.org	48.48	2.20
fr	jabber.fr	48.89	2.24
fr	im.apinc.org	48.89	2.24
fr	jabber.gnubox.net	50.43	3.33
fr	ims.kelkoo.net	45.11	5.43

fr	jabber.criirad.net	44.57	4.54
fr	e-bonnell.net	48.48	2.22
fr	jabber.nuxo.net	50.37	3.05
fr	im.drazzib.com	47.23	-1.58
gh	infonetghana.com	5.55	-0.25
it	jabber.linux.it	45.40	7.41
it	jabber.dagarlas.org	45.25	8.86
mx	jabbermx.org	25.65	-100.29
nl	jabber.ambrero.nl	52.21	4.55
nl	unstable.nl	52.35	4.89
nl	nedbsd.nl	52.30	4.90
nl	nedbsd.be	52.30	4.90
nl	shady.nl	52.30	4.90
nl	4business.nl	52.30	4.90
nl	xmpp.us	52.22	5.95
nl	12jabber.com	52.22	5.95
nl	jabber.cn	52.22	5.95
nl	jabber.anywise.com	52.22	5.95
nl	xstars.nl	52.11	5.20
nl	vanbragt.com	51.44	5.51
nl	wmijn.nl	51.44	5.51
no	jabber.no	59.91	10.81
no	manya.urbanturban.no	63.42	10.43
pl	chrome.pl	50.40	19.57
pl	jabber.wp.pl	54.22	18.38
pl	jabber.gda.pl	54.40	18.60
pl	jabber.autocom.pl	50.04	19.57
pl	remes.pl	52.30	16.40
pl	hapi.pl	51.82	19.47
pt	jabber.3gnt.org	38.79	-9.12
pt	sapo.pt	38.73	-9.14
pt	mail.telepac.pt	38.73	-9.14
pt	netcabo.pt	38.73	-9.14
pt	simplicidade.org	38.79	-9.12
pt	jabber.felisberto.net	40.20	-8.42
ro	jabber.ro	44.25	26.70
ru	volgograd.ru	48.70	44.30
ru	online.vlz.ru	49.00	44.00
ru	oborona.net	57.09	65.32
ru	jabber.ru	55.39	37.31
ru	xmpp.ru	55.39	37.31
ru	city.veganet.ru	56.33	36.72
ru	jabber.nnov.net	56.33	44.00
ru	jabber.dol.ru	55.45	37.42
ru	freeside.ru	55.45	37.42
se	jabber.cd.chalmers.se	57.69	11.98
sk	jabber.efb-group.sk	48.10	17.07

ua	jabber.od.ua	46.40	31.50
ua	lost.net.ua	40.41	30.66
ua	jabber.kiev.ua	50.19	30.53
ua	jabber.in.ua	49.99	36.21
ua	jabber.lafox.net	50.30	30.28
ua	planeta.ks.ua	46.39	32.37
ua	jabber.max.net.ua	49.58	36.20
ua	jabber.te.ua	49.33	25.36
uk	amber.org.uk	51.27	2.35
uk	jabber.ttllp.co.uk	51.50	0.02
uk	tuff.org.uk	53.97	-1.59
uk	netmindz.net	51.32	-0.50
us	jabber.org	42.23	-91.17
us	pono.mauiholm.org	20.74	-156.45
us	binaryfreedom.info	42.48	-71.15
us	bgmn.net	26.80	-80.80
us	jabber.com	39.75	-104.90
us	j20.rtcomp.us	43.41	-83.93
us	jabber.unoc.net	32.78	-96.80
us	hawkesnest.net	44.52	-89.57
us	teamthicko.com	44.52	-89.57
us	pimpstation.org	43.25	-75.98
us	sunyocc.edu	43.00	-76.20
us	malkier.net	38.88	-90.05
us	gmail.com	37.39	-122.08
us	prolly.org	39.17	-76.36
us	pyxos.net	40.22	-76.98
us	foxybanana.com	32.49	-117.07
us	inflecto.org	29.21	-98.26
us	hoffmang.com	37.46	-122.27
us	netflint.net	40.77	-111.90
us	deepdarc.com	37.45	-122.12
us	patronsystems.com	40.02	-105.25
us	ijabber.com	32.50	-97.30
us	jabberes.org	37.46	-122.26
us	jabber.vorpalcloud.org	35.71	-78.61
us	vilinkup.com	41.11	-73.80
us	stan4d.net	33.97	-118.22
us	303labs.net	33.97	-118.22
yu	www.warhammer.net	44.52	32.00
za	darkskies.za.net	-34.40	18.26
za	jabber.obsidian.co.za	-26.12	28.40