Context Awareness in Mobile Phone Based Applications Using Bluetooth

by

Jennifer Munnelly, B.Sc.

A Dissertation submitted to the University of Dublin,

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

2005

Declaration

I, the undersigned, declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Jennifer Munnelly

September 12, 2005

Permission to Lend and/or Copy

Acknowledgments

Firstly, a sincere thanks to my supervisor Dr.Siobh*á*n Clarke for her guidance, assistance and endless enthusiasm, which made this work an enjoyable experience.
Many thanks also to Cormac Driver, who willingly contributed his time, suggestions and interest.
A word of appreciation to Andrew Jackson for his invaluable help and advice throughout the year.
To all my family and friends - for their unending support. Their encouragement and patience will be appreciated always.
Finally, to the NDS boys, for making this a most enjoyable, memorable year. I hope the friendships last as long as the memories.

Jennifer Munnelly

University of Dublin, Trinity College
2005

Context Awareness in Mobile Phone Based Applications Using Bluetooth

Jennifer Munnelly
University of Dublin, Trinity College, 2005

Mobile phones have become the communication medium of choice. They have evolved into a multifaceted device capable of data services and multimedia applications in addition to their voice capabilities. Mobile phone communicative capabilities have been broadened significantly by the inclusion of technologies.

Mobile phones have the potential to be useful for more complex functionality than common voice and text usage. Applications provide capabilities that allow mobile phones to learn and use information regarding their surroundings. Such behaviour is known as 'Context Awareness'. Applications are context aware when they reason about data with knowledge of their environment. These applications may behave differently depending on the context, which tends to be dynamic in mobile scenarios. Interactions between mobile devices can be made more efficient with the use of contextual information. Applications considering context awareness on mobile phones have not been explored in great detail.

However, the evolving capabilities of mobile phones are inhibited by their limitations and constraints. The compactness of the embedded technology results in restricted resources like reduced processing power and battery life. Additionally, the mobility and geographical dispersion associated with mobile phone based applications incur common distributed systems issues. Concerns such as file storage and network partitioning are exacerbated by the mobile nature of the application and may act as inhibiting factors.

The objective of this dissertation is to examine the technological communication capabilities of mobile phones, and in particular, to investigate the suitability of Bluetooth as an underlying technology for communication using mobile phones. The ad-hoc nature of the short range wireless technology Bluetooth makes it suitable for dynamic personal area networks such as those formed by mobile phones. Bluetooth transport protocols lend themselves to the development of mobile phone applications which make use of data transfer techniques. This thesis illustrates an adaptive context aware mobile application that exploits the device discovery, service management and transport protocol functionality available in the Java API for Bluetooth Wireless Technology (JABWT). It also describes a communication layer capable of conveying contextual information on top of the Bluetooth stack.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Mobile phones have become pervasive in society, resulting in a mobile community across the globe. Ireland boasts a mobile penetration of 94 percent. With mobile phone handsets supporting new and various technologies, the area of mobile phone based applications has been broadened, with scope for more innovative and novel applications. Despite the expansion of the number of technologies available, growth in the area of phone based applications has not followed the rapid increase in handsets.

Most mobile phone users avail of the voice and SMS features of the handset, but few choose to exploit the accessible features, which offer functions varying from personal organisation and financial planning, to games and entertainment. This dissertation aims to explore the capabilities of the communicative technologies available to mobile phone users. These capabilities coupled with the inherent mobility of mobile phones offers great potential for useful application development.

Context awareness can be described as the capture and use of information about a user or object which allows them, or others affected by them, to adapt their behaviour according to the surroundings. The notion of context-awareness has become familiar in various areas of computing, and is acutely functional in mobile applications where a user's situation is constantly changing. Context awareness in mobile phones would enable the device to gain knowledge of its mobile environment. This would further add to the possible benefits mobile phone could provide to users. Studying the various forms of context in mobile environment is central to this work.

The primary objective of this dissertation is to explore the prospective benefits of incorporating context awareness in mobile phone based applications. The applications of particular interest involve communication among multiple users in an ad-hoc, P2P network using Bluetooth as the underlying technology.

1.1 Motivation

Due to the lack of useful applications developed for the mobile phone platform, questions as to factors inhibiting application development for the mobile phone platform are posed. Investigation into the ability of mobile phones to become context aware, and the constraints imposed by the mobile phone were motivating factors for this work.

This section outlines the research aims and objectives of this dissertation. The overall aim involves the investigation and evaluation of context awareness on mobile phones and to examine if and how technologies available, such as Bluetooth, are capable of implementing such applications efficiently. The descriptions below delineate the areas that had to be addressed to facilitate the end result, namely context awareness, mobile phones and Bluetooth.

1.2 Context Awareness

The exploration of context awareness within the restrictions imposed by a mobile environment was a principal objective. Effective mobile applications aim to maximize the benefit of the application use for the user. In the optimum situation, context would be non explicit input, i.e. discovered and would improve the performance and effectiveness of the application, by adapting its behaviour in accordance with the new information without requiring the user's manual intervention. This situation would require that adequate sensing equipment be available to enable the required context to be captured. As shown by recent work in the area of context aware mobile phones, outlined in detail in chapter 3, phones lack this source of embedded sensor, and even lack the ability to connect comprehensive external sensors, however impractical they may be.

1.2.1 Context Aware Computing Challenges

Mobile, context aware computing applications are those that execute on a mobile computing device and have an awareness of the user's environment e.g., location or activity status. Central to these applications is the ability to obtain accurate and useful contextual information and to exploit the information, enabling modifications in response to changes on context. The challenges of investigating what contextual information is available, identifying relevant areas, employing appropriate techniques to capture the information and finally putting the information to use, make designing and developing context aware applications challenging.

1.2.2 Mobile Environment

When building mobile, context aware applications, software developers are faced with both the challenges of context awareness and the additional complexity of the mobile environment and the difficulties associated with it. Some additional aspects relevant to this research include executing resource intensive algorithms on resource limited devices, accessing accurate and appropriate user context, providing an intuitive representation of the user's environment given the limited interface available on mobile devices and importantly, communication between devices. These challenges are exacerbated again on a mobile phone platform due to reduced processing capabilities, limited memory and battery resources, and lack of common sensor support.

1.2.3 Context in PAN

Personal area networks (PAN), such as those created when using Bluetooth enabled devices, inflict separate restraints on the forms of context available and how the contextual information may be attained. The geographical area is reduced and the signals emitted differ from other network types and influence both the acquisition and availability of contextual information.

1.2.4 Context on Mobile Phones

In addition to being within a PAN, the mobile phone handset poses its own set of variations. The phone's primary purpose of voice communication along with its rich sources of information give scope for new forms of context. This contextual information will be dynamic and plentiful, so the validity of the information may need to be considered.

1.2.5 Capture and Use

Contextual information must be attained, modelled and used correctly in order to customise an application to be context aware so that it may adapt in response to information it gathers. Mobile phones further affect the ability to encapsulate relevant context as the hardware itself limits the methods available for the capture of information. Phones lack any form of embedded sensors resulting in extremely limited discovery techniques. Investigation into whether the context may be discovered via the phone, user or environment and into the need for context to be published explicitly was also a dissertation aim.

1.3 Mobile Phones

1.3.1 Platform Limitations

The constraints associated with a mobile phone are inevitable, mainly due to its compact size. This impinges on both the technical ability of the phone, and the physical features of the handset. The recent introduction of memory cards in handsets has alleviated constraints in many areas, but the majority of handsets in typical use are still affected by some form of restriction.

Areas of investigation include:

- Reduced Storage: Storage constraints on the handset limits file volume. Mobile phones may also be restricted as to the types of files they can read, manipulate and store. Aims included investigation of how files or documents can be created and stored on the handset, and the identification of storage boundaries to decide how much data can be stored on the phone.

- Limited Processing Power: The task of challenging the limitations of the phone's ability to execute functional applications was also addressed, resulting in and evaluation of performance in processor intensive tasks.
- Constrained Battery Life: As a direct result of the reduced physical size of the phone, the battery is significantly smaller resulting in a limited duration before recharging. Determination of whether this was a limiting factor in application development and the evaluation of Bluetooth's battery consumption was an objective.
- Reduced Screen: Depicting contextual information including location requires a user interface capable of illustrating intuitive and meaningful graphics. The phone has both reduced pixel capacity and a condensed API with which to construct interfaces.

1.3.2 Communicative Capabilities

In order to effectively communicate, a device must be both capable and reliable in its ability to transfer and receive data. Using the embedded technology on the mobile phone, exploration of its suitability for communication beyond the traditional, routine use was a primary goal of this dissertation.

1.3.3 Application Development

With the emergence of new technologies, and their inclusion on the phone platform came new APIs and environments conducive to the development of applications to exploit the features and potential of the technologies. Java 2 Micro Edition (J2ME) is the version of the Java platform that is designed for use with smaller, less powerful devices such as mobile phones[17]. J2ME's place in the Java 2 Platforms is illustrated in figure 1.1.

Specific APIs exist for individual technologies e.g. JSR82 is the Java Bluetooth API suitable for mobile phone based applications. These APIs are often condensed versions of larger more comprehensive APIs and consequently result in limited functionality which can affect application development. Several integrated developement environments (IDE) and emulators exist to aid mobile application development, although not

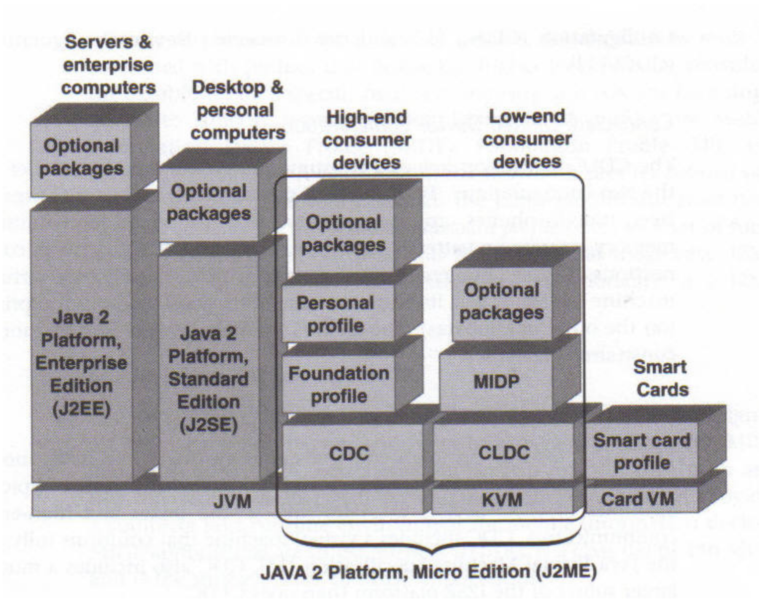

Servers & enterprise computers
Desktop & personal computers
High-end consumer devices
Low-end consumer devices
Optional packages
Optional packages
Optional packages
Optional packages
Java 2 Platform, Enterprise Edition (J2EE)
Java 2 Platform, Standard Edition (J2SE)
Personal profile
Foundation profile
MIDP
Smart Cards
CDC
CLDC
Smart card profile
JVM
KVM
Card VM
JAVA 2 Platform, Micro Edition (J2ME)

Figure 1.1: J2ME in Java 2 Platforms

all are suitable for phone based applications. It was an objective of this dissertation to observe the development process and to discern factors that may contribute to the lack of functional mobile phone based context aware applications.

1.4 Distributed Issues

Due to the distributed nature of mobile phones and the networks they create via embedded technologies, issues associated with distributed systems are inevitable and are examined as they arise. The following areas are of relevance.

1.4.1 File Storage

File creation, distribution and storage in mobile phone based applications are areas of research required to facilitate implementation. J2ME provides file functionality and the phone's native operating system software may also provide additional file access. Decisions regarding remote file access and manipulation were also researched.

1.4.2 Ad hoc network

Bluetooth networks are ad-hoc by nature. This dissertation explored the dynamic addressing and ability of nodes to join and leave ad-hoc Bluetooth networks in an attempt to discover why communicative applications are not more widespread in such networks.

1.4.3 Multiple Users

One aim of this dissertation was to assess the ability of multiple users to modify files distributed on handsets. One aspect was to determine if files could be accessed, modified and saved back to the original user's device, or whether only the user on the local device could modify the stored file. Both scenarios would raise issues of consistency across all users, creating the need for software transactionality and locking techniques or versioning and merging methods.

1.4.4 Range and coverage

With Bluetooth applications, the short range technology poses coverage problems. Communication may be interrupted by network partitions or node failure, security and authentication failures and connection failures which are common in mobile networks. This dissertation aimed to investigate mobile phone Bluetooth application range.

1.5 Thesis Outline

In this chapter, the motivating factors for this dissertation are outlined. Research aims are described and the major areas of interest defined.

Chapter 2 gives an insight into the background of each main area. This provides a more comprehensive view of the overall objective.

Chapter 3 provides an summary of the current state of the art in the areas of mobile phones, context awareness, and Bluetooth. Related work in all three areas, and more interestingly, research into the combination of all three are described. This allows this work to be put into context with regard to the current body of work in the area.

Chapter 4 describes the design stage of this work. Firstly an overview of the application developed is summarised. Then challenges that were encountered that influenced the design of the application are detailed. Finally the components and overall architecture is outlined along with any assumptions made.

Chapter 5 gives details of the implementation of the proposed context aware mobile phone based application which uses Bluetooth as an underlying technology to communicate with multiple devices in the area. Features of the application, along with the particulars of how they were implemented are included. Bluetooth functionality is explained and consistency addressed.

Chapter 6 evaluates the developed application with respect to the limitations of mobile phones.

Chapter 7 outlines conclusions drawn from the information arising from the research, implementation and evaluation stages in this dissertation. Finally areas of potential future work are identified.

Chapter 2

Background

2.1 Mobile Phones

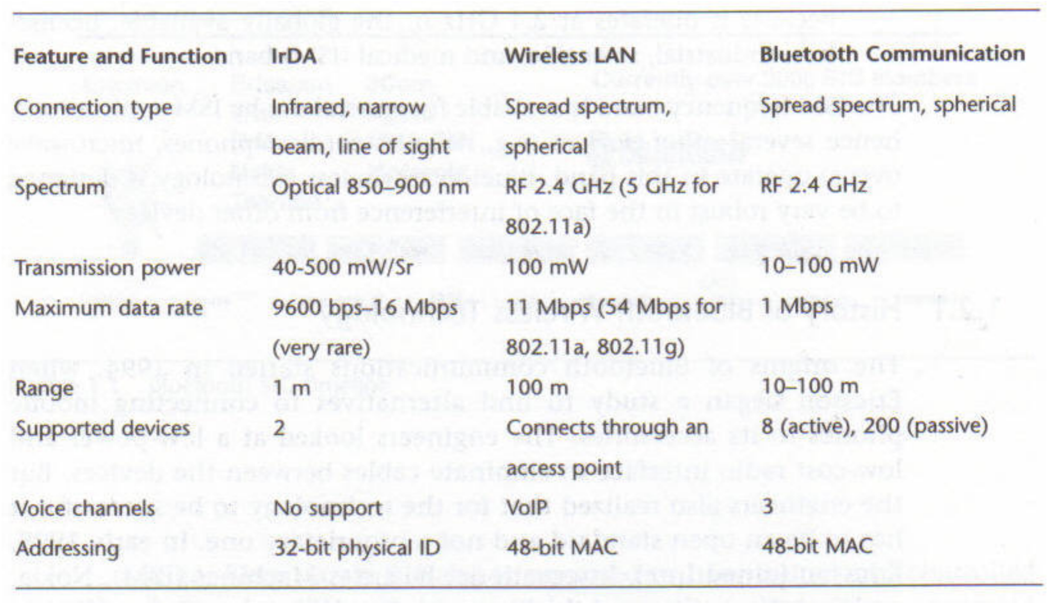
With a move towards a world in which communications and computing are ubiquitous, mobile phones are ideally positioned to be a core element in this move. With their existing widespread use and familiarity, the deployment of more complex functionality through wireless technologies has the potential to turn the conventional handset into a versatile, multipurpose piece of equipment. This section examines in detail the characteristics and areas associated with mobile phones.

2.1.1 Technologies

Wireless technologies are allowing computing and communication devices to be used virtually anywhere and to be used in innovative, progressive activities. Several of these technologies have been adapted to small devices and are now widely available mobile phone handsets, increasing the potential for new communicative operations. The technologies below are options explored when considering which wireless technology to use in the implementation of this dissertation.

GPRS

General Packet Radio Service is a packet switched data transmission protocol which is used as a mobile data service by mobile phone users [14]. It operates using TDMA



Feature and Function	IrDA	Wireless LAN	Bluetooth Communication
Connection type	Infrared, narrow beam, line of sight	Spread spectrum, spherical	Spread spectrum, spherical
Spectrum	Optical 850–900 nm	RF 2.4 GHz (5 GHz for 802.11a)	RF 2.4 GHz
Transmission power	40-500 mW/Sr	100 mW	10–100 mW
Maximum data rate	9600 bps–16 Mbps (very rare)	11 Mbps (54 Mbps for 802.11a, 802.11g)	1 Mbps
Range	1 m	100 m	10–100 m
Supported devices	2	Connects through an access point	8 (active), 200 (passive)
Voice channels	No support	VoIP	3
Addressing	32-bit physical ID	48-bit MAC	48-bit MAC

Table 2.1: Comparison of Wireless Communication Technologies

channels in Global System for Mobile Communications (GSM). GSM is a standard for mobile communication. It uses digital channels for signalling and speech. GPRS transfers data by sending packets to mobile phones over channels unused by circuit switched voice and data connections. This results in short periods of channel use only when data is being transmitted. GPRS on mobile phones is used for web browsing, email functionality and the transfer of multimedia files. Transfer rates for GPRS on mobile phones are estimated at 53.6 kb/s, but according to tests[35] "GPRS enables per-handset data rates of 9.05-107.2 Kbit/sec depending upon the coding scheme employed and time slots allocated to a data packet. In practice, transfer speeds of 400 to 1000 bytes/sec are the norm."[1]

Benefits of GPRS include

- Dependable transport system
- Geographical distribution irrelevant
- Voice calls not suspended during use

[1]Ferris Research

But it was found to be unsuitable for the objectives of this dissertation due to its limiting factors including

- Expense, cost per kb
- Extremely high battery consumption

802.11

802.11 is a wireless LAN technology. Many mobile devices such as PDAs use 802.11 for network connectivity. 802.11 or WiFi as it is commonly known, is not supported on standard mobile phones. It is available on recent devices which are a combination of PDAs and mobile phones, but is not a feasible technology for use in mobile phone based applications.

SMS

The Short Messaging Service available to all mobile phone users is a messaging system that allows users to exchange messages up to 160 alphanumeric characters long. It is a simple service usually provided by the mobile phone network operators and it operates over GSM. Once a message is sent, it is received by a Short Message Service Centre (SMSC), which must then direct it to the appropriate mobile phone. If roaming is in use, the SMSC sends a SMS Request to the home location register (HLR) to find the customer. The SMSC transfers the message in a Short Message Delivery Point-to-Point format to the serving system. The system pages the device, and if it responds, the message gets delivered. The SMSC receives verification that the message was received by the end user, then categorizes the message as 'sent' and will not attempt to send[10].

SMS does provide a reliable messaging service, and has many advantages including:

- Simple, reliable messaging
- Embedded technology enables abstraction for the user
- Guaranteed message delivery

However, it is unsuitable for the intended application implementation in this dissertation due to the following reasons:

- Cost per message
- Lack of manipulation for application development
- GSM Network delay

MMS

Multimedia messaging service (MMS) facilitates the transfer of multimedia files between mobile phones. MMS claims to 'fulfil the present and the future demands of wireless messaging[29]. Files including animations, video, audio and photographs can be exchanged using MMS services provided by the network operators. MMS makes use of common technologies and supports HTML, audio formats including MP3 and WAV and pictures including JPG, GIF and PNG. Video file format depends on the handset manufacturer, with Nokia supporting 3PG files.

MMS requires a GPRS compatible mobile phone. The main steps in MMS are outlined below

- Sender initiates a data connection that provides TCP/IP network connectivity over GPRS.
- A HTTP POST is performed to a Multimedia Messaging Service Centre (MMSC) which transfers the MMS message which has been encoded in the MMS Encapsulation format[11].
- The encoding includes a header which contains destination information.
- The MSSC performs validation on the message and sender, stores the message and creates a URL where the message is available.
- A new MMS notification message is sent to the recipient using SMS, using GPRS to initiate a data connection that providing TCP/IP network connectivity.
- The recipient performs a HTTP GET to obtain the MMS content from the URL.

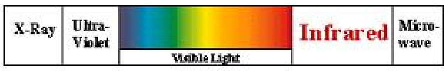


Figure 2.1: Infrared in Spectrum

MMS has obvious benefits for rich communication with its support for multiple file types, but the cost per message is so high that uptake has been slow in the SMS dominated market. The related cost and lack of manipulation techniques make it an unsuitable option for this dissertation.

Infrared

The Infra-red Data Association (IrDA) infrared communication has been introduced to mobile phone handsets in recent years. Infrared is an established technology for use in remote controls. Its context within the spectrum of waves is illustrated by figure 2.1. Standardisation by IrDA has enabled universal point and shoot infrared connectivity between devices of all types[20]. The short range wireless technology provides another option for data transfer amongst mobile phone users. The latest use of infrared on mobile phones enables users to print stored images at kiosks using a simple user interface.

Infrared is has many limitations, primarily its lack of multipoint connections and inability to work around obstacles. IrDA protocols provide connectivity range of up to one meter, and rates up to 4 Mbps, although both are in the process of being extended.

IrDA infrared is controlled by a protocol stack, organised in a familiar layered fashion, similar to the Bluetooth stack architecture.

Connections are made between two devices only, one acting as the primary and one as the secondary. The primary is responsible for link creation and management. Similarly to Bluetooth, device and service discovery techniques are employed before establishing a connection. The two participating devices will operate at their highest common transmission speed, and attempt to communicate in ways that optimize the throughput and reliability of their connection[20].

The principle benefits of infrared technology include:

- Cost free communication.
- Interoperability among various handset manufacturers.
- Easy to use.
- Low power usage.

Disadvantages include

- Line of sight requirement
- Point to point connection only
- Extremely limited range

Infrared was not found to be a suitable wireless technology for use by the mobile context aware application proposed in this dissertation. This was largely due to the lack of support for multiple users, as communication involving a group of users is a fundamental requirement.

Bluetooth

The Bluetooth wireless technology serves as a replacement of the interconnecting cables between a variety of personal devices, including mobile phones. Its aim is to function as the universal low cost, user friendly, air interface that will replace the excess of proprietary cables that people needed to carry and use to connect their personal devices. It has become the technology of choice in cordless headsets for mobile phones, and is frequently used by mobile phone users to exchange files such as ringtones and images.

It was found to be the most suitable wireless technology for this project, as it offers the benefits of omni-directionality and the elimination of the line of sight requirement of RF-based connectivity. Bluetooth creates a ”personal connectivity space” which resembles a communications bubble that follows people around and empowers them to connect their personal devices with other devices that enter the bubble[31]. Connectivity in this bubble is spontaneous and can involve several devices of diverse computing capabilities including reduced power devices such as mobile phones, unlike wireless LAN solutions that are designed for communication between devices of sufficient computing power and battery capabilities[31].

2.1.2 Platform Constraints

Size is a foremost factor in a consumer's choice of handset as smaller more compact mobile phones are easier to carry. Mobile phones have become fashion accessories with compactness being a stylish trait. The downside to miniaturisation is that technical aspects of the handsets are also made condensed. Mobile phones have restricted resources due to the compressed nature of its hardware and software. An objective of this dissertation is to explore the inhibiting characteristics of mobile phones with respect to application development (see section 1.2.1). The most obvious constraints have been outlined previously, i.e. reduced storage, limited processing power, small screen and modest battery life. The constraints detailed in this section are more specific to application development. These factors directly influence Java mobile phone based applications.

Data transfer type

Applications are limited as to the forms of data that can be transferred. Using Java Serialisation, objects may be transferred using the transport protocols of available technologies. However, it may not be possible to exchange certain file types as file support varies from one manufacturer to another. For example, Sharp phones save audio clips as AMR files while other manufacturers use WAV files. When attempting to transfer unsupported files to a mobile phone, an error is thrown and transfer is aborted.

Memory: Heap Memory, Persistent Memory

Storage is a limited resource on mobile phones. The introduction of memory cards has increased the capacity to store large volumes of data on the handset. While executing a Java application, only a certain portion of memory is available to the JVM. This is known as the 'Heap Memory'. This can be influential when applications have a requirement for large amounts of data to be stored in virtual memory.

Persistent Storage on the handset itself is made possible by using the MIDP Record Management System (RMS). The corresponding javax.microedition.rms package has a single class RecordStore. This class maintains a collection of records which are in fact just byte arrays[17]. A RecordStore created by a MIDlet is accessible by other MIDlets within the same MIDlet suite, but not by external applications. A MIDlet suite is simply a JAR file containing MIDlets and associated files. The RecordStore is created in the MIDlet by using an openRecordStore() method. This will open an existing RecordStore with the specified name, or create a new RecordStore and assign it the specified name. The RecordStore is then manipulated by getRecord() and setRecord() methods. The records are simply byte arrays with incremental IDs.

The RMS RecordStore was used in this dissertation for the implementation of persistent storage on the mobile phone, and the use of memory is explored during testing and results are described in chapter 6, Evaluation.

Maximum jar size

MIDlets require a JAR file and a JAD file in order to be deployed on mobile phone handsets. The JAR file contains class files, the manifest and any resources such as images or external files. The JAD file is a Java properties text file that contains information to allow the device execute the application[17]. The manifest in the JAR file contains similar information to the JAD file, but the JAD includes MIDlet specific information including the MIDlet-JAR-URL and JAR size.

The MIDlet-JAR-Size property gives the exact size of the corresponding JAR file in bytes. Manufacturers specify a 'Maximum Jar Size' for each device capable of running Java applications. This imposes restrictions on the number of classes and complexity of the application. This constraint has been alleviated by memory cards allowing the maximum jar size to be dynamic.

Supported APIs

Standard application programming interfaces (APIs) in the Java programming language are defined through the Java Community Process (JCP). This allows a thread of management in the evolution of the Java language. Each new API is developed as a Java Specification Request (JSR). All J2ME configurations, profiles and optional packages are defined as JSRs by the following process[17]:

- 1. A JSR is submitted with a potential specification.
- 2. The JCP executive committee reviews and votes on the JSR.
- 3. An approved JSR leads to the formation of an expert group.
- 4. The expert group define the specification.
- 5. JCP reviews the specification in a review period.
- 6. The specification is open for public review.
- 7. A final specification draft is proposed.
- 8. The JCP executive committee votes to accept or deny the API.
- 9. If accepted, the specification is released.

The supported APIs on a mobile phone define the capabilities of the device. Lack of API support limits application functionality as the technologies available for use are reduced. This point is illustrated by the difficulties outlined in section 4.3.4.

2.1.3 Interoperability

Interoperability between handsets is poor due to the large number of mobile phone manufacturers. Different makes of mobile phones are based on varying operating systems, making all native functionality OS specific. Manufacturers also vary in which file formats are supported. Even within one manufacturer, models can differ greatly. For example, Nokia uses the Symbian operating system and has multiple categories of models. An application based on a series 40 model will have varying functionality to a series 60 model, and so the application may be inhibited from running correctly if it makes use of series specific functionality.

Another aspect of interoperability that affects mobile phone applications is the inability of Symbian applications to interact with Java MIDlets. This is particularly relevant in this dissertation. Functionality is provided application development using Symbian C++ that is unavailable to MIDlets. MIDlets run within a constrained environment known as a 'sandbox'. The sandbox has a primary purpose of prohibiting installed MIDlets from accessing secure information on the mobile phone itself. Symbian applications have the ability to access this information making them very useful for the development of context aware mobile applications. This interoperability problem is significant in this dissertation and details of challenges arising from it are described in section 4.3.

2.2 Bluetooth

Bluetooth wireless technology is an open specification for a low cost, low power, short range radio technology for ad hoc wireless communication of voice or data. Its estimated over the air communication range using radio waves is 10 metres, the actual range available on mobile phones using Bluetooth is examined in chapter 6, Evaluation. Due to the short range, the radios are low power and are suited to small, compacted devices with reduced battery power. Bluetooth networks are ad-hoc by nature, and are known as personal area networks (PAN).

Technicalities of the Bluetooth Technology include:

- FHSS (Frequency Hopping Spread Spectrum).The Bluetooth radio transmission uses a packet switching protocol FHSS. The hop frequency is 1600 hops per second. The frequency spectrum is divided into 79 hops of 1 MHz bandwidth each, so devices occupy 79MHz, but at any specific moment, only 1 MHz is occupied. Frequency hopping is used to reduce interference and enhance security. The frequency-hopping scheme is combined with fast ARQ (Automatic Repeat Request), CRC (Cyclic Redundancy Check) and FEC (Forward Error Correction).

A binary radio frequency modulation and simple link layer protocols reduce the complexity and the costs of the radio chip. Bluetooth provides a nominal data rate of 1 Mbit/s[31].

- ISM Band: A Bluetooth radio operates in the 2.4GHz license-free, globally available industrial, scientific and medical (ISM) band. This band is used for various other ISM devices e.g. WLAN, microwaves, but Bluetooth is designed to withstand interference and remain almost unaffected when in contact with other devices in the same band.

2.2.1 Stack

The Bluetooth protocol stack is defined as a series of layers, with each layer representing a different protocol. The Bluetooth profiles, described along with the stack in the Bluetooth specification, are essentially usage models to illustrate how applications should use the stack. The stack can be divided into two major sections, the first is the Bluetooth host, which is the upper section of the stack and is usually implemented in software. In the case of a mobile phone, the Bluetooth host is integrated with the operating system of the phone. The lower layers are known as the Bluetooth controller. They are usually contained in a hardware element or radio module. Each of the layers in the stack is described below.

Radio

The Bluetooth radio layer is the lowest defined tier of the Bluetooth specification. It classifies the requirements of the Bluetooth transceiver device operating in the 2.4-GHz ISM band. It defines transmitter and receiver characteristics, including the ability of the receiver to measure its Received Signal Strength Indicator, which became relevant in this dissertation, see section 4.4.3 for details.

Baseband

The Baseband tier is the physical layer of the stack. The protocol is implemented as a link controller, and together with the link layer it manages the physical radio frequency link between Bluetooth devices and enables connections. The two kinds of physical

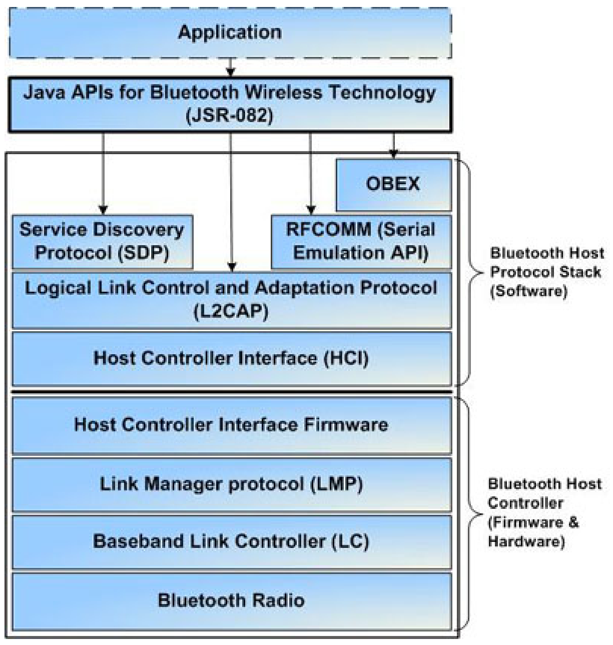

Application
Java APIs for Bluetooth Wireless Technology (JSR-082)
OBEX
Service Discovery Protocol (SDP)
RFCOMM (Serial Emulation API)
Logical Link Control and Adaptation Protocol (L2CAP)
Host Controller Interface (HCI)
Bluetooth Host Protocol Stack (Software)
Host Controller Interface Firmware
Link Manager protocol (LMP)
Baseband Link Controller (LC)
Bluetooth Radio
Bluetooth Host Controller (Firmware & Hardware)

Figure 2.2: Bluetooth Stack

links: synchronous connection oriented (SCO) and asynchronous connectionless (ACL) are managed by the Baseband which involves handling packets and the paging and enquiring techniques of Bluetooth discovery.

LMP

The Link Manager Protocol (LMP) carries out link setup and link configuration between Bluetooth devices, managing and agreeing the baseband packet sizes. Link managers communicate via the LMP using a number of PDU (Protocol Data Units), which are sent between devices to facilitate link management. The LMP is also responsible for managing security issues, such as authentication and encryption, by generating, exchanging, and checking link and encryption keys[25].

HCI

The Host Controller Interface provides a command interface to the radio, baseband controller and link manager, providing a single interface for accessing the baseband resources, the hardware status and control registers. Access to the LMP via the HCI was an area of interest in this work, and research is described in section 4.3.3.

L2CAP

The Logical Link Control and Adaptation Protocol (L2CAP) is located in the data link layer. It provides both connection-oriented and connectionless data services to upper layer protocols. It is responsible for multiplexing the various connections of the upper layer protocols. L2CAP allows higher level protocols to send and receive data packets of up to 64 kb. Communication over L2CAP is restricted to asynchronous connectionless links and so is a best effort service unsuitable for real time traffic on SCO links.

RFCOMM

One of the most frequently used communication techniques in communication devices makes use of serial ports. The RFCOMM protocol is an emulation of RS232 serial ports over L2CAP. It facilitates a transport service for higher level services using a serial interface and is capable of supporting up to 60 simultaneous links, although most devices, especially mobile phones, have limited capabilities regarding the maximum number of connections. RFCOMM provides a communication mechanism for two Bluetooth enabled endpoints, making it both suitable and feasible for this project and was chosen as the protocol of choice.

SDP

Service Discovery Protocol (SDP) provides a process for applications to query available services and attributes of services on other devices. The discovery of services is distinct from the discovery of devices in Bluetooth, and is a completely separate protocol. It is also distinct from the more traditional notion of service discovery in LAN networks as the set of services available are dynamic and change frequently when devices are active in the PAN.

OBEX

The Object Exchange protocol is a relatively new facility built over RFCOMM. It is a protocol for simple file transfers between mobile devices, often used for transferring objects such as electronic business cards[37]. It was originally implemented over IrDA, but is now common on Bluetooth devices. However the OBEX API was not supported by the devices used in this project and therefore was not investigated further.

2.2.2 Profiles

The Bluetooth specification also defines profiles which outline how a particular model may use the protocols and features. The profiles are particular to a specific use case, sometimes described as 'a vertical slice through the protocol stack'[25]. The four basic profiles are

- Generic Access Profile: All profiles are based on the GAP as it defines the discovery, link management and configuration procedures.
- Service Discovery Application Protocol: Defines core protocols used to locate services available on other Bluetooth devices.
- Serial Port Profile: Defines the process necessary for Bluetooth devices to communicate over RFCOMM.
- General Object Exchange Profile: An abstract profile for OBEX usage.

2.2.3 Use Cases

General Case

Bluetooth devices may be used in various activities with varying purposes, but they will have certain usage scenarios in common. These general use cases of Bluetooth are outlined in figure 2.3. The device may act as a client, or a server, or both. Therefore it may make use of device discovery protocols and be both the producer and consumer of services. The mobile phones in this dissertation were nodes in a peer to peer, distributed environment, and so had the capability of exerting both client and server features.

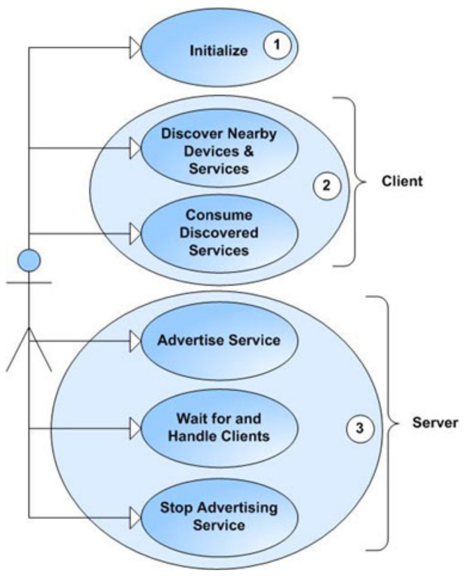

Initialize
1
Discover Nearby Devices & Services
Consume Discovered Services
2
Client
Advertise Service
Wait for and Handle Clients
Stop Advertising Service
3
Server

Figure 2.3: Bluetooth General Use Cases

An overview of the major activities of the use case scenarios illustrated by figure 2.3 are

- Initialisation: The device must initialise the Bluetooth stack.
- Client: Devices in range are discovered and then those devices are queried for services available on them.
- Server: Services are advertised to clients, making them available for query. Should any client wish to connect, the server accepts incoming connections

A more detailed analysis of how these operations function and details of how they were implemented in this dissertation are discussed in the implementation chapter, at section 5.5.

Mobile Phone Use Cases

Bluetooth on a mobile phone enables connectivity with any other Bluetooth enabled device. An example of a use case scenario for a regular mobile phone would be to extend its voice capabilities, shown in this three in one phone usage model[25]. The

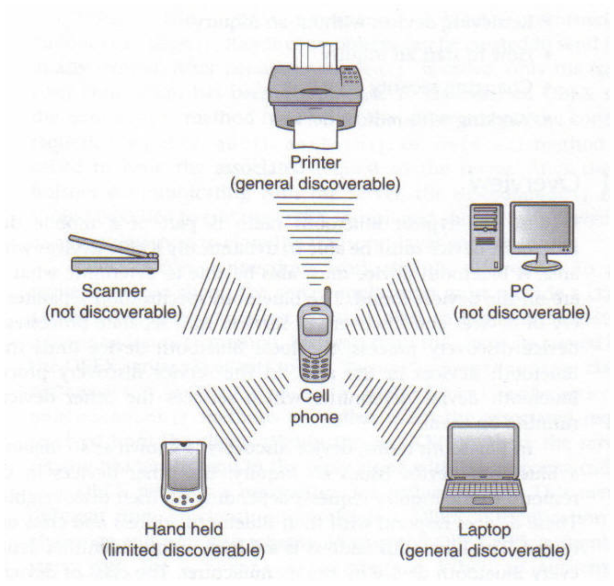


Figure 2.4: Inquiry

handset still functions as a normal mobile phone, but can act as a cordless phone by connecting to a voice access point i.e. a cordless base station, and may also act as a 'walkie talkie' by communicating directly with another similar device.
The scenario more applicable to this dissertation, is removing the need to use voice as a means of communication, and exploiting the data transfer capabilities of a Bluetooth enabled mobile phone. It can discover devices within range, and communicate directly using a messaging system built on top of the underlying Bluetooth transport layer.

2.2.4 Device Discovery

Bluetooth enabled devices use a technique called device discovery, which is core to any application using Bluetooth. It enables the device to dynamically locate other devices within the area. As device and service discovery are distinct, all devices regardless of what services they provide will be located. If a Bluetooth enabled device wishes not to be discovered, they can be configured so that they are not visible during device discovery. Device discovery is also known as an 'Inquiry'. Figure 2.4 shows a mobile phone initiating an inquiry.

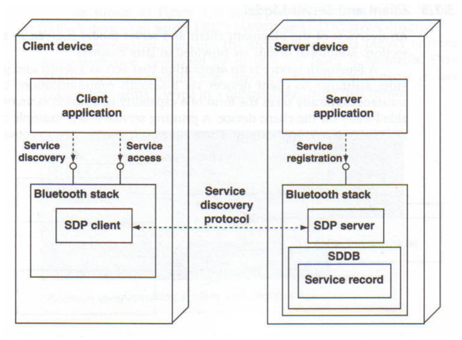


Figure 2.5: Service Discovery

A device issues inquiry requests and waits for nearby devices to respond with their corresponding Bluetooth address and device class. The address acts as a unique identifier for the discovered device and the class described the type of device.

2.2.5 Services

A service can be thought of as a particular facility that a device offers or is capable of participating in. Service functionality is performed in a client server fashion, with a server being queried by a client about its available services. Some standard services are defined by the Bluetooth specification e.g. file transfer, printing, but most application specific services are created dynamically. This section gives a brief overview as to how devices deal with offering and locating services.

Service Discovery

When an inquiry has completed and all the devices in range have been discovered, the services available on the devices may be queried before connecting to a remote device. Service discovery is conducted on a node by node basis, unlike device discovery which contacts multiple devices. Services are defined by a service record, which describes in its attributes the particulars of a service. The querying device may search for a required service using one of these attributes, and if a match is found, the remote device returns the complete service record. Service Discovery between two devices is illustrated in figure **??**.

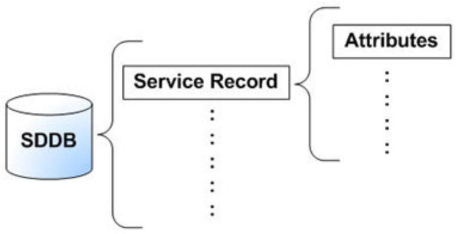


Figure 2.6: Service Discovery Database

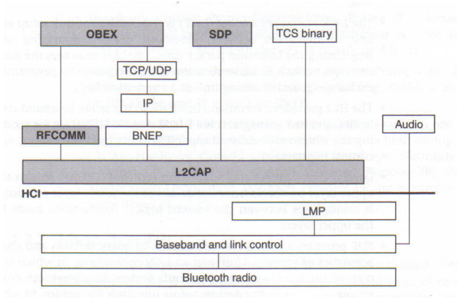


Figure 2.7: JABWT Stack

Service Advertisement

A device acting as a server maintains a service discovery database (SDDB). A service record is created describing an available service. The architecture of an SDDb is illustrated in figure 2.6. Services are registered in the SDDB in order to be available for query by remote clients. When a service request is received, the SDDB is used for service discovery. The SDDB must be updated with any changes to services and the services are removed from the SDDB when the service becomes unavailable. The details of how this is implemented using JABWT is detailed in the Implementation chapter.

2.2.6 JABWT

Java APIs for Bluetooth wireless technology (JABWT) defines the first standardised set of APIs that enable application development using Bluetooth as a wireless technology. JABWT addresses certain protocols of the Bluetooth stack, as shown in figure 2.7.

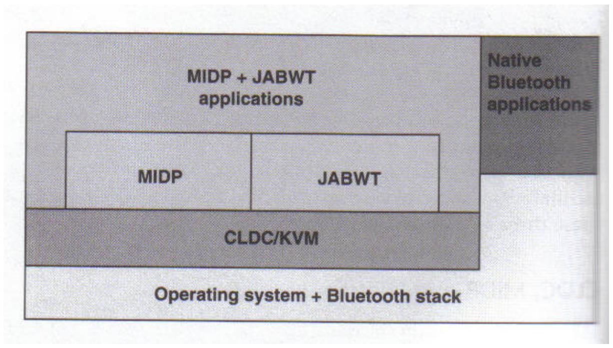

MIDP + JABWT applications
Native Bluetooth applications
MIDP
JABWT
CLDC/KVM
Operating system + Bluetooth stack

Figure 2.8: Relationship of JABWT, MIDP and CLDC

It is targeted at limited devices, such as mobile phones taking into consideration limited processing power and battery life. Most of these devices make use of J2ME, but some may use J2SE, as JABWT based the API on J2ME and the Generic Connection Framework (GCF) CLDC API. Therefore JABWT only requires a CLDC library. CLDC is usually coupled with a J2ME profile such as MIDP, but the JABWT does not depend on MIDP functionality. Figure 2.8 illustrates this relationship as it is in the implementation of this research, i.e. a MIDlet on a MIDP device supporting JABWT. It supports three transport protocols: L2CAP, RFCOMM and OBEX. It also provides basic methods for all device discovery, service registration and service discovery functionality, establishing connections over the three supported protocols and security functionality.

Chapter 3

State of the Art Review

3.1 Introduction

The state of the art aims to investigate the current situation in the areas of interest influencing this dissertation. This establishes the context in which this work will be considered. Various bodies of research are relevant, and so this section has been divided into more specific areas, although many related works will encompass several areas.

Each of the following sections gives a brief introduction to the area and its relevance to the outlined research aims. Then relevant work in that area is explored, discussing recent developments and works in progress. Finally the background research in all areas is summarised with the significant approaches isolated and the applicable areas identified.

3.2 Mobile Phones

Mobile Phones have changed the way people communicate, and various social changes have been observed due to their increased dominance. A large body of research has been carried out to review of the social consequences of mobile phone use and how everyday life is changing in relation to their use. Areas addressed include changes in living and working conditions, expectations and roles, experiences of time and space and the change in the interrelationship of public and private[16]. Studies show that time is a major factor in today's busier lifestyles, and that mobile phones and the features they provide, can change the concept of time by providing flexibility and mobility to time-shift activities. It also shows that phones are used for contact, i.e. the message content is not so much the point as the gesture. Studies have also shown that phone features such as SMS can make communication more discrete and unobtrusive to those in the area[16].

Social considerations in technologically mediated interactions are a specific area of interest. Technologically Mediated Communications (TMC), such as mobile phone activities, change many elements of communication that contribute to the social dimension of human interaction[34]. Researchers in the field acknowledge the emergence of a set of new social structures and examine the ways in which technology parameters impact on social communication. TMCs obviously provides a less rich environment for communication than face to face interaction, but this may be seen as an advantage in certain situations. The fact that mediated interactions interfere with spontaneity, creativity and the lifespan of a message can be viewed as disadvantages, but they also allow a certain amount of control over the interaction that is unattainable in on the spot conversation. Phone based applications offer a certain distance to the sender and at times a seemingly safe anonymity. It removes any communication redundancy and leaves the user with essential content, which is pre-prepared. Such is the popularity of mobile phone based communication that text messages are often used in place of voice interactions[34].

This area is fundamental to the decision of what problem to address in the implementation of a mobile phone based application in this dissertation. Considering the findings of the research in this area, the adverse issues arising from a social activity involving face to face human interaction may be alleviated by allowing the activity to be performed using a mobile phone based application.

The decision was made to implement an application that aims to provide a phone based alternative to the traditional approach to dating. It will provide a facade that a user can effectively hide behind, creating a distance between the sender and receiver which eliminates the physical, intellectual and emotional reactions together with a sense of being in control of when to terminate the communication. This added personal sanctuary is one of the major elements of technology enabled communication that can be exploited in applications such as the one implemented in this dissertation.

3.3 Context Awareness

Context Awareness has become a familiar term in computing. Many definitions can be found as to exactly what is considered to be context. One definition classifies context as any information that can be used to characterise the situation of an entity[4]. Awareness has been defined as "an understanding of the activities of others, which provides a context for your own activity"[8]. The capture and use of contextual information increases the usefulness of applications allowing them to adapt their behaviour according to their surroundings. Context has been described as usually the location, state and identity of an object or person[8], but as shown in the research studied below, context can take various forms and be interpreted in many ways.

3.3.1 Context Awareness in Mobile Applications

Wireless technologies, cellular networks, PANs, LANs and WANs, mean more dynamic context and the need for more efficient and adaptive applications in mobile environments. The increase of context aware applications ties in with the developments in ubiquitous and pervasive computing. Context has been defined in three categories - computing context, user context and physical context[3]. The use of context history can be useful in certain applications, such as trails based applications, and may allow the inference of information from contextual data. Context aware computing and its applications can be classified again into four sections: Proximate selection, automatic contextual reconfiguration, contextual information and commands and context triggered actions[3]. Although all make use of contextual, they adapt and behave in different ways in accordance with the context. Applications that are context aware may be active in that they adapt to discovered context, or passive in that they present contextual data to interested users[15].

3.3.2 Modelling Context

Modelling contextual information is an area of research that has resulted in various approaches to representing the information. Location information generally takes the form of a symbolic model, abstract symbols, coordinates or a geometric model. Due to the limited screen size on the mobile phone, location must be depicted in an easily understood format. A graphical representation is much more intuitive than coordinates, however the constructs available to create graphical interfaces are somewhat reduced on mobile phones. Other contextual information may be modelled using key-value pairs, Object Oriented models or layers of logical context[15]. Depending on the amount and variety of contextual information gathered in this dissertation, modelling techniques including name value pairs and graphical representation may be suitable for displaying the information on the limited phone screen.

The approaches of storing and disseminating contextual information are typically either a centralised server which is the provider of context, or a distributed architecture where the nodes store context, the latter being more appropriate for distributed applications such as this phone based one, but this raises distributed issues regarding storage on the limited devices.

3.3.3 Related Context Aware Work

General Mobile Context Awareness

Cyberguide is mobile context aware tour guide which has been implemented on handheld devices i.e. PDAs. The belief that all applications in a mobile environment should take full advantage of the rich contextual information that is available, and use it to offer better services to the user is central to Cyberguide[13]. Many possible scenarios for the use of context in mobile applications are outlined including personalised tours, route planning, providing directions and enhanced reality tools. The implementation uses existing hardware, namely Apple MessagePads and creates an architecture consisting of components for maps, information, positioning and communication. GPS, active badges and the Internet underlie the guide's infrastructure. The context used is very limited, just physical location and crude orientation, but the idea of using context to enhance mobile applications is relevant. It identifies the need for automated capture of contextual information to facilitate greater richness of data and to improve usability[13].

The PLIM project is concerned with the notion of context aware mobile applications and considers mobile phone applications and what context may be relevant. Mobile phones may be used for data services and may execute various multimedia applications and this research outlines that personalised and device optimised context aware communication is a vital part of emerging phone applications. Context in mobile applications is dynamic and can impact on how the user interacts with other devices. Context "is not limited to the physical world around the user but also incorporates the user's behaviour and terminal/network characteristics"[21]. They note that the majority of phone based context has been location based, yet there are multiple, more informative indicators of context awareness that combine to make up the 'context space' such as identity, spatial and temporal info, social situation, activity and resources nearby[21]. Any contextual information captured should be available to be used by the application. They illustrate this using an instant messaging system that combines and uses presence and location information to achieve a personalised and useful application. The PLIM framework uses Jabber, an XML based open source system for instant messaging. They outline location based services options such as GPS, GSM cell ID and active badges. Bluetooth serves as the core technology in PLIM, as they noted its support of ad-hoc networks and data point connectivity. This supports the viewpoint of this dissertation that Bluetooth is well suited as an underlying technology for more useful applications on mobile phones. Bluetooth discovers all devices in the area and checks if a certain service is supported using a service discovery protocol, using filters to specify results. Positioning is one service that may be provided. PLIM uses contextual information to update user details and position. Future work outlined lies in the area of coupling absolute and descriptive locations and in interpreting situational information[21].

Mobile Phone Context Awareness

'I M Mobile, Where R U?' concentrates specifically on the context awareness of mobile phone applications. It agrees that context aware applications have become more prevalent, with the continuous provision of information about presence and availability of others being commonplace in instant messaging systems online[18]. This paper explores the need for such information for mobile technology users to establish if it would improve reachability and availability. Firstly they identified which contextual information is instrumental for the communication of mobile knowledge between users. During research, they compared use of various granularities of location information such as GPS coordinates and more local, small scale information, for example 'in a meeting' or 'on the bus'. Mobile phones with integrated GPS receivers have been released, but as yet have not gained general consensus. This is thought to be due to factors such as limited awareness, cost, reliability and accuracy issues, but more interestingly privacy and security concerns. Research was carried out by experiments to gain knowledge on mobile awareness, specifically which context information people need to communicate and work together effectively. The paper focuses on instant messaging, identifying social translucence and plausible denial of presence as factors of its success[18]. It concludes from research that location revealing data is most popular contextual information, but that it is preferable from sources such as personal calendars than from GPS like devices. The conclusions from this research give useful context related principles specific to mobile phones, which most 'mobile ' applications omit due to the implementations which are generally carried out on PDAs[18].

More mobile phone specific context aware work has been carried out with the intention of making informative context available to the mobile phone user. Two papers in particular address a particular area of interest, namely the capture and representation of contextual information on mobile phones. The first is titled 'SenSay: A Context Aware Mobile Phone'[7]. It addressed the lack of sensor support on mobile phones. The approach taken was to mount external sensors on the person, and to connect these to the mobile phone. Sensors including accelerometers, microphones and light sensors. These monitor the user's actions and environment and aim to provide data about the user's context. The data is passed through a rule based decision module and a resulting action is taken is direct response to the discovered contextual information. The objective of this work was to create a 'context aware mobile phone'. Context is discovered and adaptive behaviour enabled, but the mobile phone itself is still unable to conduct this behaviour without the use of various peripheral attachments.

Extending the functionality of the phonebook on a mobile phone is the primary objective of the second work. 'Context Phonebook'[2] aimed to extend the phonebook by including context with the contact details. Relevant context identified included users connection status, availability and location. Like the previous work, the lack of sensor support was discovered to be a particular problem. Rather than adding external sensors, an approach was taken to try to keep functionality within the handset. The implementation relied heavily on manual settings and simulations.

From the research in this area, numerous approaches to the identification, capture and representation of context are described. The approach of using external sensors is not only impractical, but expensive and time consuming due to the various vendor specific hardware involved. Interoperability issues are inevitable, as are high setup costs and substantial setup time. For these reasons the use of external sensors was established to be an unfeasible approach. The approach taken in [2] involving manual and simulated context setting is a more feasible method and the implementation stage of this dissertation was influenced by these findings.

3.4 Bluetooth

For the purpose of this dissertation, Bluetooth was chosen as the most practical wireless technology available on mobile phone. It is described in detail in section 2.4, but defining it as a 'short range wireless technology, suitable for low-power, low-cost applications" clarifies its core characteristics. It is more flexible and robust than its competitors such as Infrared. Each Bluetooth device is identified by an IEEE MAC Bluetooth address. Bluetooth is controlled by a set of protocols known as the Bluetooth stack, see section 2.4.1. Nodes act either as 'masters' or 'slaves', which are theoretically similar to the notion of 'server' and 'client'. Masters have more authority and accept connections from up to seven slaves to make a piconet. Piconets may be conjoined to make scatternets, although scatternet implementation in this dissertation is not viable due to the lack of support on phones. This dissertation gives each node the functionality of acting as both server and client, forming an interconnected network topology.

3.4.1 Bluetooth and Java

Java has become the language of choice for a large fraction of software developers. Its portability and mobility make it ideal for wireless application development and deployment. The Java platform is being driven forward by emerging technologies, to make it a comprehensive development environment. Some of these technologies are targeted at wireless, short range, P2P and ad-hoc network applications. Bluetooth is well positioned to be used as the underlying communication technology for the implementation of these applications. Therefore Java and Bluetooth have converged to allow the development of such applications.

J2ME, a cut down version of the Java class libraries, had been released to enable development on devices with limited processing power. The Connected Limited Device Configuration (CLDC) is a J2ME library that targets mobile devices with small memory and wireless connectivity[17]. The J2ME Mobile Information Device Profile (MIDP) provides a set of user interface components and HTTP connection capabilities for use in mobile phones and other mobile devices. J2ME is explored in detail in section 5.1.2. The Java Specification Requests (JSR)-82 Expert Group is responsible for the standard Bluetooth APIs for the Java platform. These are known as JABWT and are described in section 2.4.7. The APIs are aimed at the CLDC and take the form of a J2ME profile. Having a standardised API for Bluetooth application development via Java means that the principles of portability, mobility and interoperability are maintained by all developers[23]. Adhering to a standard API ensures interoperability at all levels of the Bluetooth stack from baseband upwards. Some Java Bluetooth stacks do not fulfil JSR-82 such as Harald and JBlueZ. However J2ME has an optional compliant Bluetooth API, and others exist too including Rococo's Impronto and BlueJava. BlueZ is the official Linux Bluetooth stack written in C++[22].

3.4.2 Related Bluetooth Work

"JXTA over Bluetooth" is the title of a thesis which focuses on the idea of developing and implementing an ad-hoc messaging system using Bluetooth. The Java APIs for Bluetooth were used to develop a multi-hop messaging layer which sits over the Bluetooth stack. The title is quite misleading as it is not an actual implementation of JXTA over Bluetooth. This thesis chose to develop a proprietary messaging system instead of using JXTA or JXME. JXTA uses XML to transmit over TCP/IP and uses far too much processing power for small devices and JXME needed a JXTA proxy and so could not implement an ad-hoc P2P system independently. Instead a system based on JXME, using the same messaging format was created which carried out connection establishment, connection management and data processing. A rudimentary computer based diary application was created and run on iPaqs to illustrate the technical mechanisms[22].

3.5 Communication Applications

Systems are rarely used in isolation and so interaction between multiple users is an element of almost all applications in one way or another. Distributed systems, especially ones with dynamic mobile nodes, must consider multiple users, synchronisation issues and any other factors arising from communication and data transfer in a distributed environment.

3.5.1 Related Communication Applications Work

The following works are all in the broad area of applications facilitating multiple users to communicate and interact. Various approaches have been investigated and are outlined below.

MobiShare is an architecture that was designed to provide a middleware system and framework that acts as a distribution network by offering ubiquitous connectivity to mobile devices[9]. The architecture allows for mobile devices to publish, discover and access resources over a large area using 802.11 wireless network. The implementation is based on XML like languages and protocols which offers a service oriented system, i.e. each device is considered a producer or consumer of data wrapped as information services. Unfortunately like so many 'mobile' applications this research was implemented on PDA devices so lacks mobile phone specific information. However, context awareness is included in this approach and is described as an essential feature in environments where objects are constantly moving, creating dynamic contextual information. The properties identified as context are used to help the mobile devices be aware of its immediate surroundings and to adapt as necessary. The data considered included position, orientation, lighting, conditions and identity[9], but implementations only used location and mobility parameters. MobiShare is implemented using a decentralised backbone network, the Internet or a WAN, providing centralised access points in communication cells known as CAS(Cell Administration Servers) for the mobile nodes. The CASs are responsible for registering the devices services and enabling other devices to discover available services in the area. Although not mobile phone specific, MobiShare illustrated the use of context and resource sharing in mobile devices.

Research has been carried out to investigate methods for providing group awareness to participants of e-meetings using small, limited devices such as mobile phones. The work titled 'Supporting E-Meetings on Java Capable Mobile Phones' aims to overcome the limitations of the mobile phone platform and to create a collaborative environment for activities involving multiple users. Observing the increase in mobile phone capabilities regarding running custom software and the changes in wireless technology resulting in more pervasive, faster networks[36] the potential for innovative communication over mobile phones is evident. This work aimed to exploit the communication capabilities of mobile phones while considering context in much the same way as this dissertation. An existing online E-meeting system known as 'Marratech Pro' was extended to suit this experiment. They endeavoured to reduce the existing system to deliver just enough features to give a sense of group awareness while running within the confines on the limited platform[36]. The existing features included instant messaging, audio capabilities, shared whiteboard, shared browser and video support. The major limiting factors of the mobile phones were found to be the screen size, limited memory resources, limited HTTP operations supported in MIDP and the lack of APIs for accessing and modifying the digital still images and video clips that phones are now capable of capturing. The effects of these limitations led to a cut down communication layer which was only capable of unicast communication. The images required also had to be scaled and scrolled to be displayed on the phone screen. The implementation consisted of an instant messaging server which is contacted by the mobile device via a Java plug-in for e-meeting data. The plugin collects context about activities carried out from the data retrieved by the pull mode the application operates. The shared whiteboard is implemented by the client requesting a scaled down version of the whiteboard page. GPRS is used to operate the shared browser. No audio support was possible but video snapshots were retrieved by the plug-in and sent to the client. Contextual information was limited to simply CPU activity and event monitoring, and this information could be displayed illustrating activities and times using graphs. The system offered a scaled down PC system with very limited context. It was a basic centralised GPRS pull model system that allowed documents to be viewed on mobile devices, but lacked any innovation and useful contextual information. This research identifies phone platform limitations in environments involving multiple users interacting. The findings are useful to this dissertation, but the centralised approach which uses fixed PC's for data administration is an entirely different architecture from the intended mobile phone based application.

MIRES (Mobile Information Resource Exchange System) focuses on mobile phone interaction and like the above works, aims to build a framework that encourages communication and data transfer[16]. The main objective of MIRES is to facilitate phone users to share personal resources and to access others resources. The paper acknowledges the increased potential of mobile phones due to the inclusion of more wireless technologies and multimedia capabilities in recent times. GPRS and J2ME were found to have facilitated developers allowing data transmissions via http. Limited storage was still a major issue, as was low bandwidth and high cost. To lessen these limiting factors, a dedicated framework was built to implement flexible resource sharing among users. MIRES regards each phone as an autonomous computing client. It adopts a centralised mobile resource management system that acts as data sharing manager[26]. The decision to have a centralised controller was mostly to overcome the issue of lack of storage on the client side as well as the slower processor and limited bandwidth. MIRES has 3 main components; a global management system (GMS), multiple mobile resource database (MRDB) servers and a bulletin board[26]. It describes its architecture as a totally distributed system which consists of several decentralised database servers. The MRDBs are physically distributed. When users want to share resources, they upload them to the server side and others can download as they like. Transmission between phones and MRDBs is implemented by SMS, for notifications, and GPRS, for data transmissions. The GMS is responsible for controlling and coordinating the underlying resources. The bulletin board holds all the indices of the resources. Each mobile has a mobile mirror image (MMI) stored in the MRDB which offers storage for that user's resources and is identified by the phone number. Resources are shared by means of "exporting" resources to be shared and "importing" required resources, which are carried out using a Java plug-in on the phone. Resources are accessed using commands including R-EDIT, V-EDIT, DISCARD and PURGE[26]. Distributed systems issues arise and algorithms are employed to handle version control and to maintain consistency via synchronisation. Problems encountered were J2ME limitations, resource loss due to lack of use, data relocation and security. This paper illustrates that while limitations are inhibiting factors, that communication and data transfer can be carried out successfully over mobile phones.

The next related work ties together the mobile environment, multi-user communication and messaging over Bluetooth, addressing a relevant area. 'Smart Service Architecture for Small Devices'[12] presents a middleware platform, suitable to small devices that allows runtime reconfiguration and adaptation on applications running on these small devices, typically mobile phones[12]. The adaptive middleware supports the discovery, downloading and activating of functionality via a standardised infrastructure. The platform makes use of JXTAs messaging format, not JXTA implementation, and uses J2ME MIDlets to deploy the applications. The primary component is an adaption manager which supports the discovery of services and their use. The architecture was used to implement an application which would trigger an SMS service once the phone was in a specific Bluetooth hotspot range. Jadabs[19] a dynamic architecture for lightweight ad-hoc networks was used for laptop and PDA communication with the phones. Jadabs uses a Service Oriented Architecture to develop components independently, and Aspect Oriented Programming, so that registered services can be dynamically adapted at runtime by having method calls on crosscutting services using AOP advice calls[12]. The Service manager is provided as the core Midlet suite, and provides discovery mechanism and the ability to download new services. This manager is registered in Bluetooth and is accessible through its Service Discovery Protocol. Once a service manager is found, the JXME discovery mechanism is used for any other service discovery. These managers use JXME messaging to communicate. Downloading can be carried out using either over the air (OTA) transport, i.e. HTTP over GPRS or WAP, or using the Object Exchange Protocol (OBEX) over Bluetooth. OBEX allows push and pull from devices using Bluetooth and was not available at the time of this implementation. OBEX offers data exchange capabilities beyond simple text but is not supported by the hardware available for the implementation of this dissertation, and so was not explored further.

3.6 Summary

The various elements that feature in this dissertation have been investigated along with the related research in those areas. Reviewing the work existing in each of these areas helps to identify the gap in the current body of research, which this dissertation aims to address. Various approaches to similar projects are outlined, and elements of these, along with an original approach, combine to present the architecture adopted by this dissertation.

Chapter 4

Design

This chapter outlines the design stage of the mobile context aware application implemented in this dissertation. Firstly the scenario which the application aims to address was formulated, and the requirements identified. In order to establish the full design and structure of the application there were two major areas to consider, messaging and context.

- Messaging requires a communication layer to be built over an underlying transport system. Bluetooth had been decided as the most suitable wireless technology, so potential messaging methods were examined to identify suitability. The most significant possibility was JXME. This is examined in detail in section 4.2.

- To enable an application to be context aware, meaningful contextual information must be identified. Methods of capture must also be considered. The various challenges encountered while undertaking the task of including context are outlined in section 4.3.

The findings from investigation into these areas with respect to the application requirements for this dissertation formed the basis for the design of the said application. Various hardware problems were also encountered, causing additional modifications to the design of the application.

4.1 Dating Application

The application decided upon was a contemporary adaptation of a dating scenario, taking certain requirements from the scenario such as speed dating. The traditional approach would involve numerous people meeting in a confined meeting place, such as a function room of an establishment. They would each sit at a table and talk to the person opposite for a period of time before moving onto the next person.

4.1.1 Social Considerations

Having researched the social interests regarding mobile phones, they offer an innovative approach to such a scenario. Removing the need for face-to-face dealings and replacing them with messages via phones, grants a significant amount of control of the situation to the user. The medium, in this case the phone, is simply a facilitator of the conveyance of an expressive activity by the expresser to the recipient[34] and the user has the ability to interact with all or one user in particular. More insight into technology enabled communication is described detail in section 3.2 with respect to research carried out in the area.

4.1.2 Requirements

The application must be capable of providing a similar service to that of a speed dating service. Users must have the ability to see all other participants and communicate freely with them at any time. An additional requirement added was the ability to privately communicate with a chosen participant. During the application execution, the user can modify certain aspects of their contextual information that is available to others, and query context features of other members in order to gain insight into their environment.

4.2 JXTA/JXME

JXME was considered as a possibility for use as a communication layer over Bluetooth. JXME is simply a small footprint of the JXTA environment that is suitable for application development for smaller devices including mobile phones.

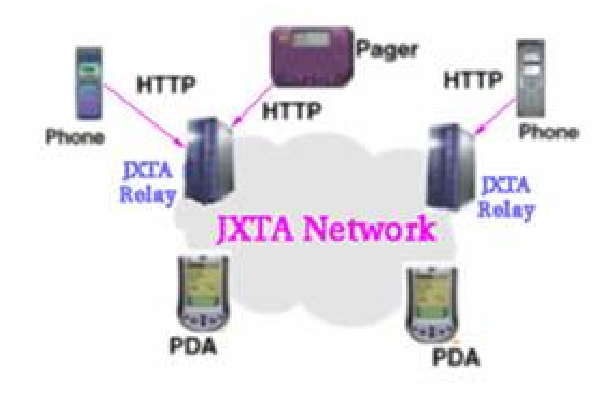


Figure 4.1: JXTA Network

JXTA was initiated by Sun Microsystems to enable further P2P application development in Java. The advantages of P2P systems have become obvious with scalability being a driving factor. JXTA aims to provide a set of simple, small and flexible mechanisms to support P2P computing on various platforms. The basic architecture provides six protocols defined in XML. Nodes implementing any of these protocols are referred to as peers. Data, known as messages, are transferred via Pipes which are virtual communication channels. Peers offer services to other peers in which Codats, content, can be published, discovered and replicated as required. The architecture of a JXTA network is illustrated by figure 4.1.

The six protocols that underlie the JXTA project are

- Peer Discovery Protocol -used to discover all published resources.
- Peer Resolver Protocol -used to locate some service or content.
- Peer Information protocol -used to ping a peer.
- Peer Membership Protocol -used to join or leave peer groups.
- Pipe Binding Protocol -peers use pipes to access services.
- Endpoint Routing Protocol -used to route messages to a destination.

JXTA has a Java implementation in the .net.jxta.* and .net.jxta.impl.* hierarchies. A JXTA shell is a sample application that acts similarly to a Unix Shell, providing access to the JXTA environment[24]. However JXTA has high memory and CPU requirements and is therefore inapplicable to small devices such as mobile phones[22]

The protocols provided by JXME would enable an application running on mobile phones to contact other devices and transfer messages effortlessly using the standard JXTA messaging format. The application could also make use of the group functionality JXME offers. These factors made JXME a suitable messaging system to implement for this dissertation.

To make JXME a viable option, it would need to use Bluetooth as the underlying transport protocol, as opposed to its traditional HTTP use. This area was explored in some detail to establish if JXME could be transformed to run over Bluetooth.

4.2.1 JXME and Bluetooth

JXTA is suitable to be implemented over Bluetooth. Recent research states that "A device using JXTA over Bluetooth could discover other local JXTA devices and form ad hoc, P2P networks with them to achieve some common, collaborative goal, or to share information and services"[23].
The Jadabs project[19] produced an API using Bluetooth as the transport technology for JXTA. Jadabs added an additional transport layer for Bluetooth. Bluetooth uses an RFCOMM link to communicate between devices. The usual Bluetooth master, slave roles exist in piconets but extended scatternets are not yet implemented by mobile phones. A JXME peer can be a normal peer or a rendezvous peer. All normal peers connected to a rendezvous peer are in the same piconet. The standard Bluetooth service registration and discovery functions are employed and an RFCOMM connection is established when the StreamConnection is created on both sides[1].

The task of implementing JXME over Bluetooth was attempted in[22]. The main handicap of JXME is its need to have a JXTA proxy, which makes an ad-hoc usage of JXME impossible since this requires a JXTA proxy server and TCP/IP network infrastructure. Two devices that are places next to each other do not communicate directly. To allow such a direct communication both devices need a HTTP server. This is too heavy weight for such a small device. These HTTP servers are required since such a message system communicates asynchronously[22].

JXME have recently released a proxyless version with no previous implementation over Bluetooth. The initial design intended to use Bluetooth to discover devices, and to use JXME functionality for service discovery and messaging format. Unfortunately, proxyless JXME was found to be unsuitable for use on mobile phones. This is due to the fact that the proxyless version, as yet, only supports devices using CDC, not CLDC, eliminating mobile phones. Attempts at partial implementation were also prevented by the lack of a Windows compatible version of proxyless JXME[33].

4.3 Contextual Information

The state of the art in context awareness on mobile phones shows that the familiar forms of context used in the wider computing community, are also the most relevant in mobile environments. Having considered the time limitations, this dissertation aimed to include context forms which hold a large amount of significance. These may have more potential from which other users may extract meaning. The context areas included

- Identity.
- Location/Proximity.
- Availability.

These primary areas, along with an attempt to obtain contextual information from the phones native features, were investigated to provide a final design for implementation.

4.3.1 Identity

The ability to recognise a user or object is central to interaction between devices. With communication being a core objective of the proposed application, having a form of identity available to all users was essential. Due to the use of Bluetooth as the underlying technology, the notion of identity becomes more complex. Bluetooth enabled devices are considered nodes within the network, and are identified by a unique identifier known as the UUID, Universal Unique Identifier. The node is also distinguishable by its unique Bluetooth address. Although these allow identification of the node in a technical sense, these forms of identity are not user friendly or intuitive.

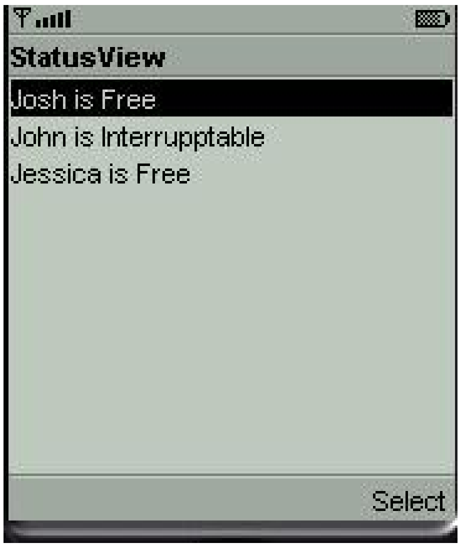
Users need to be recognisable by an understandable name or label. Bluetooth uses an attribute 'Friendly Name' which allows a user to assign a chosen name to a Bluetooth device. This does not uniquely identify the device, but is more user friendly. For example, a user may decide to assign 'Josh's GX25' as the friendly name. This name will be displayed when other users view devices during discovery.

To further develop the forms of identity available to users in the application, a username attribute is associated with each participating user. The user enters their chosen username on launching the application. This username, along with the device identifiers, is used throughout the application to identify users during activities.

4.3.2 Activity Status

Users can make decisions based on information about other user's situations. If a user is known to be actively engaged in as conversation, knowledge of this could allow other users to adapt their behaviour accordingly. This notion of 'Availability' was considered to be an important piece of context within this dissertation. Availability context has been included in attempts to create a context aware mobile phone in recent research[2]. The knowledge of a user's availability status can influence a decision on initiating communication and so should be made available to all other users.

The question of how to determine a user's activity status was a key issue. Due to the lack of sensor support on mobile phones, automated discovery of availability is impossible without the use of external sensors. The addition of external sensory devices is an approach that facilities automated context discovery[7], but is highly impractical. The most feasible option was to allow the user to set their activity context manually.


StatusView

Josh is Free

John is Interrupptable

Jessica is Free

Figure 4.2: Representation of Activity Status

This approach has been used for availability context[2] and proves to be an adequate solution.

To represent the activity status of a user, the easiest form of representation was a visual display of all usernames and corresponding statuses. The display is shown in figure 4.2. The user must have the ability to set and update their activity status at any time during participation, and all nodes must be notified of updates.

4.3.3 Proximity/Location

In an environment with multiple users interacting, knowledge of location can be of great benefit. Information attaining to the users whereabouts can be used to create points, maps and trails. These creations can be displayed in a visual manner to users, which is a more intuitive form of representation than text.

Location sensing devices and technologies have been used to determine mobile location including GPS and active badges[21]. These methods are more suited to PDA like devices and are not supported by standard mobile phone handsets. Bluetooth provides a method of determining approximate location in two basic ways. The first is an inherit knowledge that if a device is discoverable then it must be within range, meaning it is in an area of approximately 10 metres of the inquiring device.

The second is by means of the received signal strength indicator (RSSI). The Link Manager Protocol Layer of the Bluetooth stack contains both the hardware and software functionality involving RSSI. The LMP obtains the RSSI value from devices so that it can control the power output to the device. This is a power saving function, but the RSSI is a direct reflection of the devices proximity to the querying LMP. The reason the term proximity has replaced location in this context is because RSSI gives no indication of orientation. This makes the option of depicting accurate location impossible, but proximity relative to the querying device is still possible.

The 'Golden Receive Power Range'(GRPR) is the optimum range for RSSI values. The RSSI value is measured against the top and bottom limit of the GRPR, but an algorithm can be employed to get a more accurate proximity reading.

The LMP is accessible only through the HCI. HCI commands are available to access lower level layers and values within those layers. The majority of Bluetooth development is done using Linux, and the HCI is much more accessible on that platform. There are numerous examples of attempts at using RSSI for location including the Atlantis location based services project[38] which is patent pending as it details a new algorithm for calculating location using RSSI. Location context resulting from RSSI has been used for context awareness in multiplayer games over Bluetooth[27]. A strong argument for RSSI use in proximity is made in a paper regarding proximity as a security property in mobile enterprise application context[5]. The idea of using past RSSI values to track device movement has been implemented in an ubiquitous infrastructure[6].

The Microsoft Bluetooth stack, which is embedded in the Windows Service Pack 2 (WSP2) installed on the laptop in use for this dissertation, has a HCI interface which supports some LMP access functions. However it does not support a command to access the RSSI. Using the Microsoft SDK it may be possible to construct a command that can contact remote devices and query RSSI, although during research only unconvincing attempts in C++ were found.

Existing applications Cellspotting[2] and CellTrack allow mobile phone users to view GSM cell information. The granularity of location information provided is ideally suited to short range networks and the information holds great potential for use in context aware applications.
However, these applications are the phones native OS, Symbian applications. Interoperability between Symbian applications and MIDLets is not possible, and so the incorporation of the data in these existing applications was not feasible.

There is no support whatsoever for RSSI exchange between mobile phones. This coupled with the existing challenges of accessing the mobile phone RSSI from a laptop and data exchange from mobile phone to laptop (see section 4.4.2), the decision was taken to include a simulator for RSSI values in the design for this dissertation.

4.3.4 Calendar Items

Another form of context that would be beneficial to the application proposed in this dissertation is information about events set as calendar entries. Mobile phones have features such as contact lists, calendars and calculators as standard. The calendar allows users to set reminders including birthdays, meetings and other dates of importance.

The objective was to investigate if the information stored in the native functions of the mobile phone were accessible by installed applications running on the phone. In particular to this dissertation, access to the users own birthday could enable filtering of users on an age bracket basis. Access to calendar functionality from within the MIDlet would also add an option of entering a new event if two users agree a rendezvous date.

As the application is developed on a Series 60 Nokia phone, the Series 60 Platform Calendar APIs[30] are available for application development purposes. However, these are limited to Symbian application development. To obtain calendar information as

[2] www.cellspotting.com

context would require a new development environment setup and development in Symbian C++. The resulting application would again encounter the interoperability issues which prevent Symbian applications from interacting with MIDlets.

Nokia have released a Java SDK which lets developers extend MIDLets beyond the sandbox model which restricts access to native data on the phone. This SDK enables access to the user's calendar, contacts and other data, including information from other MIDlets. This is done via a File Connection API and the related Personal Information Management (PIM) API. This functionality has only been released recently and is supported by PDA devices and some mobile phones with extended functionality, but was not supported by the devices available in this dissertation.

Ultimately, this area of context was not feasible for inclusion in the final design.

4.4 Hardware Challenges

4.4.1 Bluetooth Dongle

To make a PC/Laptop Bluetooth enabled, an external Bluetooth adapter, commonly known as a Bluetooth dongle can be attached. For the purpose of this dissertation a Belkin Bluetooth USB Adapter was used. This enabled the transfer of MIDlets from the laptop to the mobile phones.

Windows has its own Bluetooth settings, and the device was not recognised as a Bluetooth device after installation with the vendor specific drivers. In order to enable Windows to recognise the device, it was necessary to reinstall the adapter allowing Windows to choose their own drivers, bypassing the vendor's software.

4.4.2 Laptop Connectivity

Bluetooth enabled mobile phones have the capacity to transfer files with any other Bluetooth enabled device, providing it also supports data transfer. Beyond the limited functions provided by the phone, i.e. image and audio transfer, applications must be developed to allow more complex interactions.

MIDlets are developed and tested in development environments which act as emulators. Emulators recreate the scenario of using a real mobile phone. The Wireless Toolkit (WTK22) is JSR82 compliant and so is able to emulate a Bluetooth stack to run MIDlets that make use of Bluetooth on phones. Ideally, if it was possible to plug a real Bluetooth stack into the emulator the MIDlet running on the phone might be able to communicate with the MIDlet running in the emulator. This is not possible so a specific application must be created to run on the laptop using the real embedded Bluetooth stack.

A bridge is then required between the laptop Bluetooth hardware and Java applications. One solution to this is Bluecove[32] an open source implementation of the JSR-82 Bluetooth API for Java which supports the Windows XP SP2 Bluetooth stack. This was installed and a serial port application using RFCOMM connections used to initiate communication between the laptop and mobile phone. Initially this did not function as intended, but this was due to an issue on the particular mobile phone handset(see section 4.4.3). With that issue rectified, communication over RFCOMM between an application and a MIDlet was successful. Due to time restrictions, the task of converting the full functionality in the MIDlet to an application for the inclusion of the laptop was omitted but could be considered as future work.

4.4.3 Nokia 7610

The mobile phone available for use in this dissertation initially was a Nokia 7610 smartphone. The specification had all the required elements namely:

- MIDP 2.0
- CLDC 1.0
- JSR82 Bluetooth API

As only one mobile phone was available for use for a period of time, development of the context aware mobile application using RFCOMM in Bluetooth was carried out using emulators. When deployed onto the Nokia 7610 the application appeared to execute in an identical fashion to the emulator without communication with other devices.

The first attempt of communication between mobile phones was using the Nokia 7610 and a Nokia 6600, which has an identical specification regarding the essential elements outlined above. The MIDlet was installed on both devices. Initialisation, device discovery and service registration and discovery executed without difficulty. However, data exchanged between the devices seemed to only work in one direction and was often interrupted by an unexplained IOException. The underlying problem could not be identified. Various attempts were made to rectify the difficulties including firmware updates and code refactoring, with no effect.

The same experiment was carried out using two identical Nokia 7610 handsets. This situation resulted in no exchanged data being displayed at either end of the Bluetooth connection. With further investigation into the transfer of data using RFCOMM, an emulation of serial ports using Bluetooth, it was apparent that the Nokia 7610 has a fault in its ability to communicate using this Bluetooth transfer protocol. Nokia included a note in their recent 'Known Issues' publication on the inability of the handset to correctly execute applications using InputStream and OutputStream classes using the serial port profile.

Due to this hardware issue, the application development required a decision as to what course of action to take.

- Port the entire application to use L2CAP instead of RFCOMM. This option had been previously implemented by developers in the Nokia Developers Forum experiencing the same problem. Various difficulties were reported including Symbian OS errors.
- Use alternative handsets, ensuring JSR82 compatibility.

The option of using other handsets was both less time consuming and more reliable, and so Nokia 6600 handsets were used in place of the Nokia 7610. The MIDlets executed without difficulty and communication between handsets occurred over an RFCOMM Bluetooth connection.

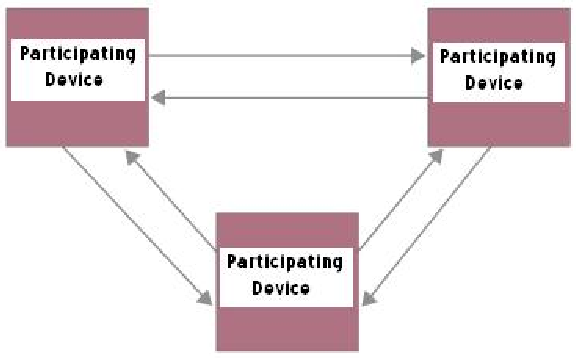


Figure 4.3: Bluetooth Application Topology

4.5 Architecture

As this application works on a peer to peer basis, each device is capable of acting as both a server and a client. Each instance of the application firstly behaves a client on start up, searching for existing servers and joining in. It can then behave as a server and accept incoming join requests from new clients. Figure 4.4 illustrates how devices running the application may be connected on the Bluetooth network.

4.6 Components

The application will be developed as a MIDlet using the Java APIs for Bluetooth Wireless Technology. The package javax.bluetooth provides the various classes needed for Bluetooth functionality. Figure 4.5 shows the Components available to a MIDlet using JABWT.

4.7 Assumptions

Certain assumptions apply to the implementation of this application. These are mostly due to the complexities which arise in given circumstances, and due to time restrictions are necessary in order to address the core elements of the application.

The first is that the service unique in its area, i.e. there are not two available

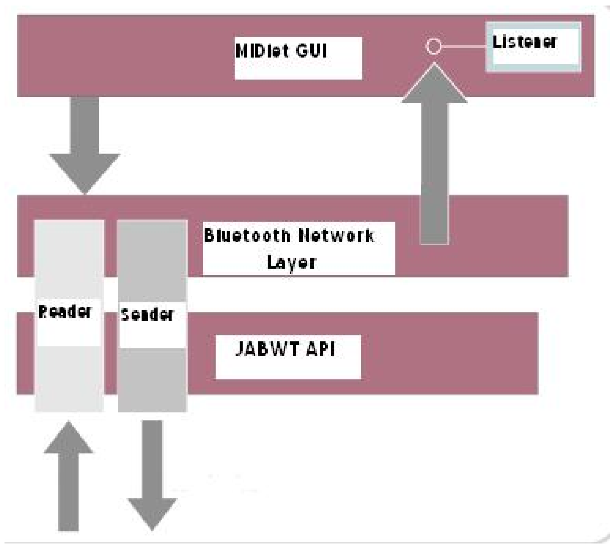


Figure 4.4: Bluetooth Network Communication

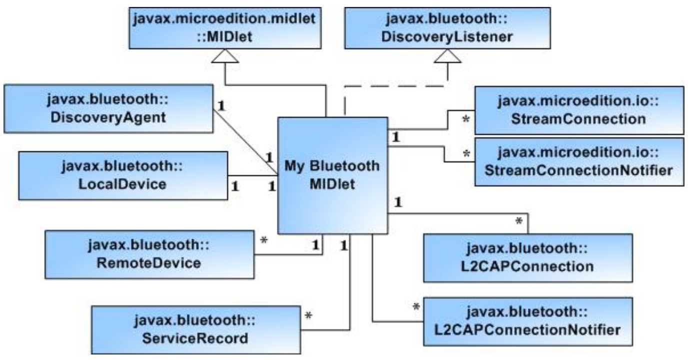

javax.microedition.midlet ::MIDlet
javax.bluetooth:: DiscoveryListener
javax.bluetooth:: DiscoveryAgent
javax.microedition.io:: StreamConnection
javax.microedition.io:: StreamConnectionNotifier
javax.bluetooth:: LocalDevice
My Bluetooth MIDlet
javax.bluetooth:: RemoteDevice
javax.bluetooth:: L2CAPConnection
javax.bluetooth:: ServiceRecord
javax.bluetooth:: L2CAPConnectionNotifier
1
*

Figure 4.5: Bluetooth MIDlet using JABWT

services with the same UUID and name. The second is that each mobile phone runs one version of the application at any one time, and the final one, is that authentication is not necessary for devices to communicate. Security is configurable through JABWT, but was unnecessary and time consuming, and can be considered as future work.

Chapter 5

Implementation

This chapter describes the work carried out during the implementation phase of the project. The core activity during this phase was the construction of an application suitable for execution on mobile phones, which was essentially an implementation of communication over Bluetooth with the inclusion of the context forms identified as feasible earlier in the project. This was to demonstrate the findings formulated from the research into context awareness in mobile phones and to illustrate them from a real world perspective.

The major objectives of implementation were:

- Explore mobile phone application development process
- Implement communication over Bluetooth
- Deploy applications on handsets
- Capture and use relevant context

5.1 Mobile Application Development

5.1.1 Development Environment

Developing MIDlets requires a slightly different development environment from that used to create standalone Java applications. To facilitate the development of MIDlets emulators recreate the mobile phone environment and allow for similar execution on regular PCs. Figure 5.1 shows the graphical recreation of a mobile phone in emulation. J2ME forms the basis for mobile phone based applications and MIDlets must be developed in a J2ME environment.

Sun's J2ME Wireless Toolkit (WTK) is the standard emulator used for MIDlet development. It contains a minimal development environment, a mobile phone emulator and some demo applications as examples. As the WTK does not offer a text editor another environment is required if full IDE functionality is preferred. The WTK simply compiles source code from a specified directory and creates the appropriate JAD, JAR and manifest files required in a MIDlet suite. It also has the functionality to run the created MIDlet on a mobile phone emulator. The emulator is a GUI similar to a real mobile phone. WTK version 2.2 supports JSR82 Bluetooth API and so is suitable for the implementation section of this dissertation. JBuilder offers a full development environment, but incorporating the Bluetooth APIs was troublesome, and so development was conducted in JBuilder transferring source files to WTK for recompilation and execution on the emulator.

5.1.2 MIDlet

A MIDlet is a Mobile Information Device Profile (MIDP) application. It is managed by 'special-purpose application-management software (AMS)[28] that is embedded within the mobile phone, but the application itself controls the detailed workings of the MIDLet.

MIDlet life cycle

The class responsible for the main MIDlet functionality, BTManager in this case, extends javax.microedition.midlet.MIDlet and implements the three core life cycle methods which are : startApp(), pauseApp(), and destroyApp(). There are three possible states in a MIDlet's life-cycle:

- paused: The MIDlet instance has been constructed and is inactive.
- active: The MIDlet is active.

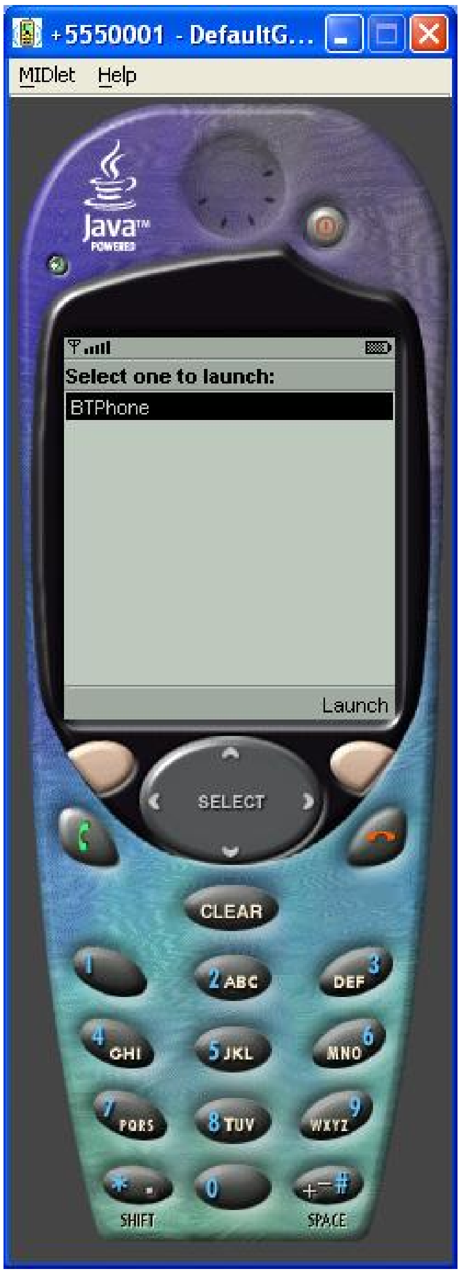


Figure 5.1: Sun Wireless Toolkit Mobile Phone Emulator

- destroyed: The MIDlet has been terminated and is ready for reclamation by the garbage collector.

After construction the MIDlet startApp() method is invoked and it displays the user interface as instructed by the application. The destroyApp() method is invoked on termination of the application or on exiting via an error.

The MIDlet needs additional files in order to be deployed on an actual mobile phone. The application descriptor is a text file that outlines details of the MIDlet suite. A manifest .mf file is also required as the standard JAR manifest packaged with the MIDlet suite.

5.2 Existing Bluetooth Code

Bluetooth application development for mobile phones is still a relatively new area. Numerous websites and forums exist to assist developers in their attempts at Bluetooth application development. This dissertation makes use of examples from various sources. Ben Hui[3] has compiled a useful source of resources for Bluetooth application development which was found to be both helpful and useful. Explanations of areas including device discovery, service discovery and RFCOMM and L2CAP communication are available with corresponding code examples. These provide a good foundation on which to build Bluetooth applications. The architecture of the application implemented is illustrated by the UML Class Diagram in figure 5.2.

5.3 Features

This section gives an overview of the features available to the user when participating in the dating service. The technical details of how each element was implemented are explained comprehensively in the following sections.

5.3.1 Messaging

The application offers two distinct forms of messaging to the user:

[3]www.benhui.net

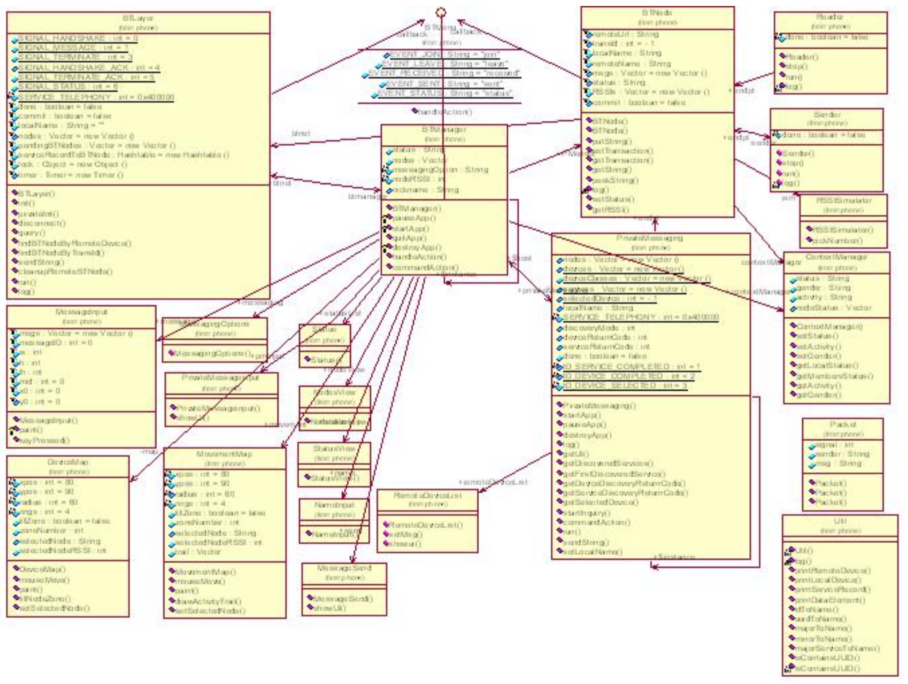


Figure 5.2: Dating Application Class Diagram

- Global Messaging: This form of messaging facilitates the transfer of messages to all active devices, or users. Once the user chooses to join the Global Message Board, Bluetooth mechanisms are employed and the user is given a connection to each device within discovery area, providing they are participating in the application.

- Private Messaging: A user can choose to communicate using a Private Messaging function which opens a distinct connection with a particular user. Again Bluetooth mechanisms are used to display all the devices in range participating in the dating application activity, and the user is free to choose the required member from the list.

5.3.2 Activity Status

One form of contextual information that is key in an application in these circumstances, is an indication of the user's availability. Without the use of external sensors or measurement devices, the task of automatically setting the availability of a user becomes a near impossibility. Therefore the user is given a menu option to manually set their status at will. The choices available are:

- Free: The user is available to communicate freely with other users.

- Busy: The user may be engaged in Private Messaging with another user or may not wish to be distracted from their current activity.

- Interruptible: The user may be currently communicating privately, but still wants to be seen to be available to other users.

To add an element of implicit activity discovery, the status may be changed by the application on an event based basis. This would set users to be automatically 'Busy' once in a private conversation and the inverse to 'Free' when in global communication again.

5.3.3 View Member Status

This option allows users to query the activity status of each participating member. A list is constructed of each user in range and their respective activity status, which is available to the requesting user. This enables them to make informed decisions regarding communication with any particular user.

5.3.4 View Member Proximity

Location is a principal form of context and is useful in all context aware applications. As discussed in section 4.4.3, precise location details were unobtainable between Bluetooth enabled phones, and so a simulation of a technique used to infer proximity was implemented to demonstrate the use of location context in an application such as this one. RSSI values are randomly generated on demand, one for each user request of proximity. The details of how this was implemented and how the RSSI value dictates the approximate proximity of the remote device is detailed in section 5.8. The proximity is then illustrated on the users screen by means of an image depicting the distance from the requesting user.

5.3.5 View Member Trail

This option is an extension of 'View Member Proximity'. A user maintains a record of the proximity of any user from which they requested location information. This builds up to generate a notion of a trail describing the physical movements of the user. For example, if a user requests location information from another user three times, the trail may show that the remote user moved closer for a period of time, and then retreated to a much further distance. This is again displayed by means of an illustration on the user's mobile phone screen.

5.4 Launching the MIDlet

The BTManager class extends MIDLet and acts as the controlling factor for the entire application. It implements the core methods startApp, pauseApp and destroyApp. The MIDlet initialises itself the first time its startApp() method is invoked. The BTManager class has two other major functions. It implements CommandListener which is responsible for handling all of the events arising from user inputs. The commandAction method is composed so that it reacts to each event appropriately. BTManager also implements an interface BTMenu, specifies a handleAction method to deal with Bluetooth specific actions including joining, leaving and receiving messages.

The constructor of BTManager initialises all of the GUI components required throughout the application. It also creates a new RecordStore from the java.microedition.rms package, detailed in section 2.1.2, in order to allow persistent storage of messages on each user's phone.

The startApp method is called on launching the MIDLet. BTManager gets a reference to the unique instance of a Display object, which is created for the MIDlet. This object is used to manage the display and input method, i.e. the phone's menu and buttons, throughout the lifecycle of the MIDlet. In order to change the display to the user, the diplay.setCurrent() method is passed an object that is an instance of the Displayable class such as a canvas or text field. In this case on launching the MIDlet the display is set to NameInput class.

The NameInput class extends Form from the javax.microedition.lcdui class, which is inherently Displayable. It consists of a TextField with instructions, and the user is shown a screen in which they fill in the username they will be identifiable as during the application. The setCommandListener method is used to set BTManager as responsible for handling the event that is created when the user hits the 'Ok' command that has been added to the phone menu.

BTManager recognises the event, and sets the display to the MessagingOptions screen, which extends List. Messaging Options, like NameInput and all other Displayable objects also sets BTManager as the command listener. Two items are added to the list: Global Message Board, and Private Messaging. The event fired from the selection of one of these results, although in slightly different forms, initiates Bluetooth connectivity.

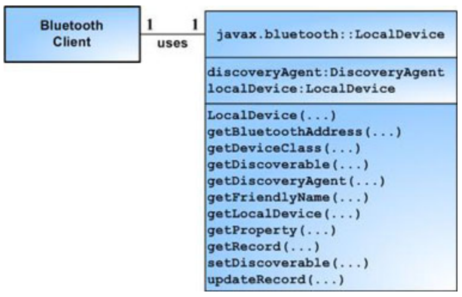


Figure 5.3: Bluetooth LocalDevice Class

5.5 Bluetooth Connectivity

This section provides details of the technical implementation of the requirements and features described. The JABWT was used to implement Bluetooth connectivity, choosing to use the RFCOMM protocol, the emulation of serial ports, as the communication protocol due to both its extensive use in current Bluetooth applications, and its suitability to the devices involved. Each of the sections below explain how this application and its classes use Bluetooth to carry out connectivity, context capture and use and communication. They follow the Global Messaging path of Bluetooth Connectivity, and the technical difference between Global and Private Messaging are described in subsection 5.5.6.

5.5.1 Initialisation and Device Management

On selection of joining the Global Message Board, BTManager creates an instance of the BTLayer class. This class is responsible for the Bluetooth networking layer and performs Bluetooth actions using the JABWT delivering the core Bluetooth functionality of the application, including device and service discovery. On construction, BTLayer is passed a reference to the BTManager, as it implements the BTMenu for handling Bluetooth events.

BTManager then calls BTLayers init() method, to initialise the Bluetooth stack. LocalDevice.getLocalDevice() returns a singleton object representing the underlying device. The LocalDevice class is illustrated in figure 5.3. This will only be obtained once, and enables the application to access more Bluetooth features via the API. It has a unique Bluetooth address which acts like a MAC address. The localDevice discovery mode is set by calling setDiscoverable(GIAC). This parameter, General Inquiry Access Code, means that the device will be available for discovery by all other devices in an inquiry. The DiscoveryAgent object is obtained by calling localDevice.getDiscoveryAgent(). This DiscoveryAgent object will be used to initiate device discovery at a later point.

The BTLayer then starts its own thread, and the run() method performs the functionality to act as a server, to register and advertise the service, and eventually accept client connections.

Each node is created as a Bluetooth Endpoint with its own connection and ability to accept in coming connection requests from clients. Service Discovery Protocol- An SDP Server maintains a Service Discovery Database (SDDB) of service records that describe the services on the local device. Remote clients can use the SDDB to query an SDP server for any service records of interest. A service record provides sufficient information to allow an SDP client to connect to the Bluetooth service on the SDP server's device.

5.5.2 Server/Service Advertisement

A server connection object of type StreamConnectionNotifier is created and will be used to handle client connections. The server connection requires a serial port profile URL, a Unique Universal Identifier (UUID) and a service name. The URL starts with btspp:localhost, which is required if you're going to use the Bluetooth Serial Port Profile which this application does. The UUID is a custom identifier for the service and can be any bit sequence unique to that service. The service name is a recognisable string that defines a name for the service entry.

The code below shows the creation of the server connection:
Server = (StreamConectionNotifier)Connector.open("btspp://localhost:" +uuid.toString() + ";name = Bluetooth Message Board");

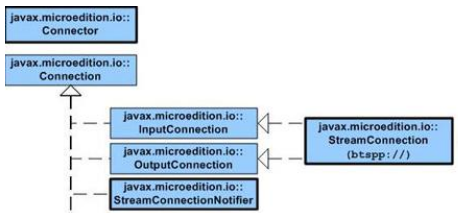


Figure 5.4: Bluetooth RFCOMM Connection Framework

Figure 5.4 illustrates the connection framework used to create the connection.

For services to be discoverable by remote devices, they must be advertised by the server on which they are located. This service registration takes place in the BTLayer class. A ServiceRecord object describing the service has been created and is obtained by calling getRecord() on the local device passing the StreamConnectionNotifier as a parameter. The ServiceRecord returned then has its availability set to indicate that the service is accessible by all other devices. The Bluetooth specification has a set of defined Service Classes which can be added as an attribute to ServiceRecord to give other users information about the service provided. This service is set to 'Telephony', to indicate a mobile phone based service.

A StreamConnection is the class used to facilitate connectivity between devices. The server.acceptAndOpen() method is called, indicating that the server is ready to accept client connections. By calling this method, the ServiceRecord object is registered in the Service Discovery Database (SDDB), ready to be queried by any potential client. The lifecycle of a ServiceRecord is illustrated in figure 5.5 including creation, storage in SDDB, and removal from SDDB.

The server device thread waits for any incoming connection requests, however at the same time, the BTManager calls the query() method on the BTLayer, initiating device discovery. This is required as there may already be an existing Bluetooth network available with servers to connect to.

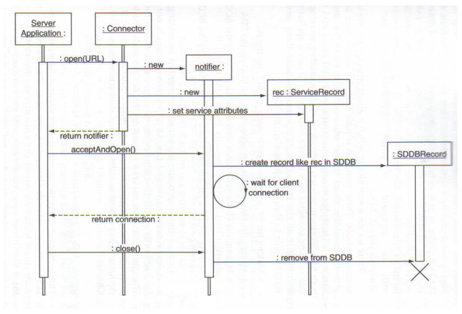


Figure 5.5: Service Record Lifecycle

5.5.3 Device Discovery

Device discovery is initiated by the query() method in BTLayer. This simply calls startInquiry() on the previously obtained DiscoveryAgent object. BTLayer has an innerclass Listener, which implements javax.Bluetooth.DiscoveryListener. This handles device and service discovery events. The deviceDiscovered() method is invoked when the inquiry finds a device, and a RemoteDevice object is passed to the handler. BTNode is a wrapper class for the RemoteDevice, allowing attributes and behaviour to be specified and associated with a mobile phone using the application.

The RemoteDevice is used to create an inactive BTNode, and it is added to a vector of pending nodes. The reason the nodes are not activated at this stage is that they may or may not provide the service required. All Bluetooth enabled devices will be found at this device discovery stage, and so redundant devices will be identified in the Service Discovery stage. The inquiryCompleted() method of the DiscoveyListener is invoked when all devices in the area have been discovered and put on the pending vector. Service Discovery is the next stage of Bluetooth connectivity.

5.5.4 Service Discovery

Service Discovery is conducted in an innerclass DoServiceDiscovey in BTLayer. It again uses the JABWT DiscoveryAgent object for discovery and invokes searchServices(). The service required is specified by passing the method the UUID we specified

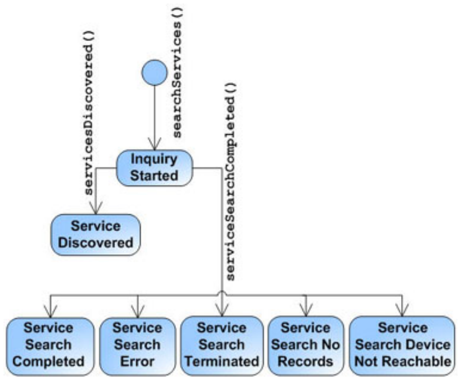


Figure 5.6: Service Discovery State Chart

earlier in regard to our Message Board service. Service Discovery is one node by node, and so for each device on the pending vector, searchService() is called. If the required service is found on the device, the BTNode wrapper attribute of transactionId is set to the return value of searchServices() to enable identification of the RemoteDevice at a later stage. A service discovery will also trigger the servicesDiscovered() method in the Listener, which uses the transactionID to get the BTNode the service has been discovered on. The suitable device is then put in a Hashtable with its corresponding record, and will be dealt with when service discovery is complete. When all pending nodes have been queried for the service, the pending vector is cleared and the serviceSearchCompleted() handler of the DiscoveryListener is invoked.

ServicesSearchCompleted() will be triggered in a variety of scenarios. These scenarios are illustrated in the state chart in figure5.6. Either service discovery completed without finding any records, having found records, by throwing an error or the service discovery may have been terminated by an external factor, as in this case a phone call on the mobile phone. In the case of services being found, the ServiceRecord object is retrieved from the Hashtable and a URL obtained by calling ServiceRecord.getConnectionURL(). This URL is passed as an argument so that a StreamConnection is established e.g. StreamConnection connection = (StreamConnection)Connector.open(url); The RemoteDevice which offers the service is made into an active BTNode. The StreamConnection is set in the BTNode and it is added to a vector of active nodes which the BTLayer manages. This vector will be used later in various circumstances as it holds references to all participating devices known to the LocalDevice. Two threads, Sender and Reader are created and started for the new BTNode. Sender is responsible for the distribution of messages, as it controls access to writing to the OutputStream. The Reader thread receives data from the InputStream. Once the BTNode physical connection has been configured, a handshake message is sent to exchange names.

5.5.5 Connection

The initial handshake is received by devices via the Reader thread. The Reader sets the RemoteDevice name and sends a handshake acknowledgement back. It also calls the BTManager to handle a 'Join' Bluetooth event, which confirms the new user's participation as a BTNode.

5.5.6 Private Messaging

The major difference between the above described Global Messaging and Private messaging is that the service discovery is limited to one chosen device, and not carried out automatically to connect to all participating devices within the area. On choosing Private Messaging from the MessagingOptions screen, a Private Messaging object is created. This initialises exactly the same as the Global Message Board. The startInquiry() method is called similarly, and is handled again by an innerclass Listener which is a DiscoveryListener implementation. When inquiry has completed, a Displayable object is populated with all the RemoteDevice objects found, and the screen is set to display this list. This allows the user to choose the particular node that they wish to connect to independently. The BTManager acts once again as a CommandListener for events in the PrivateMessaging class. A StreamConnection is established as described and communication is conducted via an InputStream and OutputStream controlled by the Sender and Reader threads associated with the two BTNodes involved.

5.6 Messaging

Once BTNodes have been setup and have established connections, the screen is set to MessageInput, which extends javax.microedition.lcdui.Canvas. The menu options include 'Add Message'. This option displays another screen, the MessageSend displayable object which is a TextBox enabling message creation. The user creates the message and triggers an event by choosing 'Save' in the menu options. The BTManager commandAction() method handles the event. The message is retrieved from the MessageSend TextBox and is sent via BTLayer's sendString() method.

BTLayer traverses the vector of known devices, and calls putString() on each BTNode, passing the message and a signal to indicate that a message is enclosed. In each BTNode, the putString() method adds the message to a vector of pending messages and notifies the Sender thread that a message is pending. Each Sender retrieves the message and writes the message to the OutputStream.

Back in the originating node, BTManager continues its handling of the event and the entered message is then converted to a byte array. It is converted to this format so that is can be added to the RecordStore, where it can be stored with all the other messages and displayed to the user on screen.

In each of the BTNodes which were sent the message, the Reader thread reads in the data from the InputStream and causes an event to be triggered, which is again handled by the BTManager. It adds the message to the receiving BTNode's RecordStore as described above. In all devices, including the originating device, the MessageInput canvas is repainted and shows the message on screen.

5.7 Status

The 'Change Status Context' option in the user menu causes the display to be set to StatusView. This extends List and has three activity options for the user to choose from: Free, Busy and Interruptible. The user traverses the list and chooses 'Select' from the Menu to trigger a handler in BTManager. Each BTNode has a ContextManager class associated with it, which performs functionality pertaining to the setting, storage, and use of contextual information. The ContextManager retrieves the chosen activity status by using the index of the chosen list item to obtain the item itself. BTManager then sends a signal, STATUSSIGNAL, to each node in its vector of active BTNodes, alerting them to the change in its status. The Reader thread in each node receives the signal and again calls on its respective BTManager to handle the event produced. The handleAction() method is passed the status that has been chosen and the BTNode that sent the message, so it sets the status in its reference to that device to the chosen activity status.

'View Member Status' in the main menu, triggers the BTManager commandAction() to create a new StatusView object, which is a displayable class extending List. It passes the vector of active BTNodes to its ContextManager, who in turn traverses the vector, querying each BTNode for its activity status, and populating another vector with the BTNode name and status. This is returned to the BTManager, who appends each node and status to the StatusView list and eventually allows the user view the context information on screen by calling display.setCurrent(StatusView).

5.8 Proximity

The 'View Member Proximity' option in the mobile phone menu enables a device to view the simulated proximity of another BTNode or device. Each BTNode has a class, RSSISimulator, which is responsible for generating random values within 10 places either side of the GPR, details of how RSSI values operate are outlined in section 4.3.3. The choosing of this option causes the BTManager commandAction() to create a new NodeView object, which is again displayable extending List. The NodeView object is populated with all active BTNodes representing devices that are visible to the querying device. This is displayed to the screen, allowing the user to chose the user/device/BTNode whose proximity they require to view.

When the user selects a BTNode from NodeView, BTManager compares the selected device to all the active devices it knows. When it matches, it calls getRSSI() on the BTNode instance chosen. The BTNode has created an RSSISimulator and calls pickNumber() on the object, obtaining a randomly generated value between -10 and +10. RSSISimulator uses the java.lang.Math.Random class to generate these values. Before the BTNode returns the generated RSSI value, it adds it to a vector which will be used to keep track of all the simulated movements of the device.

The RSSI integer is returned to the BTManager where its value examined to identify how close the RSSI value indicates the RemoteDevice to be. Because no orientation information is available, zones of proximity are used to indicate location relative to the querying device. A DeviceMap object is created as the graphical interface on which proximity is depicted. The zone to be depicted on the DeviceMap must first be identified. The following code shows how the RSSI values correspond to 'zones'.

```
if(nodeRSSI<=-5)
     map.fillNodeZone(4);
 else if(-4<=nodeRSSI && nodeRSSI<=0)
     map.fillNodeZone(3);
 else if(1<=nodeRSSI && nodeRSSI<=5)
     map.fillNodeZone(2);
 else if(6<=nodeRSSI && nodeRSSI<=10)
     map.fillNodeZone(1);
 map.setSelectedNode(selectedNode, nodeRSSI);
     display.setCurrent(map);
```

DeviceMap is a canvas and using the limited functionality of javaz.microedition.lcdui.Graphics, a map resembling a target board is created illustrating the areas immediately surrounding the LocalDevice. The number of zones and a set increase in radius are used to create the image. The BTManager, as shown in the code above identifies which zone the remote BTNode is located in and calls fillZone() on the DeviceMap passing the zone number as an argument. This sets a boolean flag in the DeviceMap and when it is being painted, the flag indicates which zone needs to be filled. The same algorithm is used to create the filled zones as the empty zones. An example of how the second inner circular space would be filled is illustrated by the code below, where the radius is a fixed integer representing the radius of the outside ring, and xpos and ypos are the

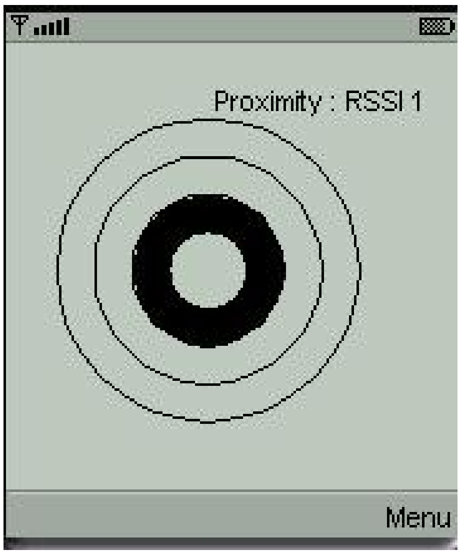


Figure 5.7: Proximity Representation

fixed starting x and y values for the image.

```
else if(zoneNumber==2){
   int rad2 = radius - radius/rings*2;
   int rad1 = radius - radius/rings*3;
   g.fillArc(xpos-rad2, ypos-rad2, rad2*2, rad2*2, 0, 360);
   g.setColor(255,255,255);
   g.fillArc(xpos-rad1, ypos-rad1, rad1*2, rad1*2, 0, 360);
   g.setColor(0,0,0);
}
```

This is displayed on the screen and is shown in a working example in figure 5.7.

The menu on the View Member Proximity screen includes the option of 'View member Trail'. This allows the user to view the movements of that particular BTNode, inferring contextual information. A MovementMap displayable canvas is created for the purpose of illustrating a BTNode's trail. BTManager is once again called upon to handle the event created when the user selects the 'View Member Trail' option from the menu. The BTNode in question is identified and its vector of recent RSSI values is obtained. This is in turn passed to the MovementMap object by calling

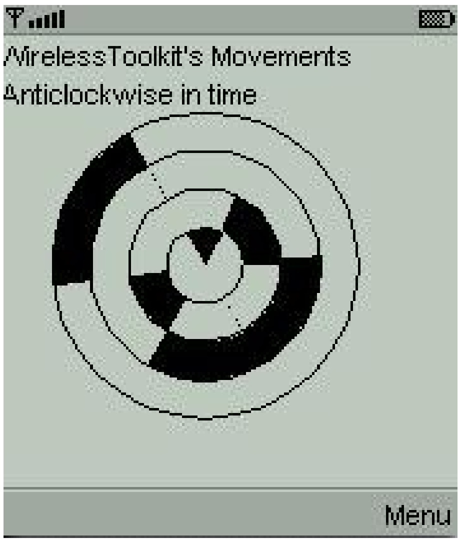


Figure 5.8: User Trail by Movement

MovementMap.draawActivityTrail(tempNode.RSSIs). The vector of RSSI values is traversed in the MovementMap class. Having investigated various display options, the most intuitive way of illustrating proximity and movement over time that was chosen was to use the same zone architecture, as proximity is always relative in this case, not absolute, but to divide the movements into arcs of the full circle, with each section illustrating the position of the RemoteDevice for a period of time. Figure 5.8 shows how this was displayed on the mobile phone screen.

This was implemented by adding a start and finish degree value to the fillArc() method, and was repeated for all values in the BTNode RSSI vector.

5.9 Consistency

In any system, consistency must be maintained to ensure all users have the same view and information. This is easily accomplished using a centralised architecture where one part of the system holds all information required by the users. In a distributed system information may be dispersed amongst several or all nodes. The maintenance of consistency becomes a more complex task.

In the mobile context aware application in this dissertation, a fully distributed, P2P ad-hoc network is formed by mobile phones. With contextual information being set and stored locally in each node, nodes must update all others to ensure a uniform view of the application state.

Multiple threads running in an application where activities update the same object simultaneously may result in problems such as race conditions and deadlock. Ensuring the application is kept thread safe is critical to its correct execution.

Two techniques were employed to implement consistency and ensure thread safety, namely critical sections and transactionality.

5.9.1 Critical Sections

Java synchronisation allows each instance of and object to have a lock, and each class to have a lock for its static methods. If a thread enters a method, for a particular instance, marked with the 'synchronized' keyword or if the thread enters a clock of code contained in a block delineated by synchronized(Instance), then the thread picks up the instance's lock[17]. When the lock is being held by a thread, no other thread can enter a method or synchronised block for that instance until the holding thread exits the synchronised method or block. This prevents two threads from updating or using data which may be in the process of being modified.

The methods wait() and notify() can be called from within a synchronised block or method, ensuring to synchronise on the same object that wait() or notify() is called on, e.g.

```
     synchronized (sender) {
             sender.notify();
 }
```

Deadlock occurs when two or more threads are waiting on locks held by one another, resulting in an infinite cycle of waiting.

Synchronisation is included in the implementation of this dissertation, however the need for synchronisation is questionable on mobile devices which use CLDC, i.e. mobile phones. This is due to the thread scheduling on such small devices. In most implementations of CLDC the single processor switches from one thread to another only when the currently active thread waits or finishes. This means that even infinite loops need to wait occasionally and none of the race conditions or deadlock situations should ever create problems even if synchronisation isn't correct[17].

5.9.2 Transactionality

A transaction is an execution of a unit of work that must be executed atomically, that is completed together or not at all. The ACID properties define how to ensure a transaction is completed properly, they are

- Atomicity
- Consistency
- Isolation
- Durability

The mobile context aware application proposed has distributed contextual information. Users are free to update certain properties used as context. To maintain consistency across the entire application, each and every node must be notified of the updated context. Distributed transactions require all nodes to make the same decision about whether to store the updated context or not. Transactionality was implemented by enforcing a two-phase commit like protocol when updating nodes. A boolean flag named 'commit' is created and set to false in the node which has changes its context. Each other node also has a boolean value called 'transaction' which is also set to false. When the updated node informs each other node of the new information, it sets the 'transaction' flag in that node to true. After confirming that all nodes have been notified of the new data by checking the 'transaction' flag in each node, the 'commit' flag is set to true. The nodes wait until the 'commit' flag is true and then store the updated context. If any nodes 'transaction' value is still false after all nodes should have been informed, updated node is aware that the transaction has not completed and can take appropriate action to ensure each node does not save the inconsistent data.

Chapter 6

Evaluation

This chapter seeks to evaluate the measurable affects of the limiting factors outlined in sections 1.3.1 and 2.1.2 on the implemented application.

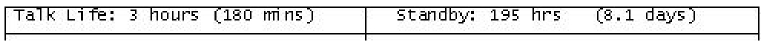
6.1 Testing

6.1.1 Test Environment

The testing was performed in a test environment with as little interference as possible, i.e. no 802.11 activity was taking place. Two Nokia 6600 handsets were used to conduct the tests. Identical MIDlets were installed on each device. The mobile phones had full battery life on commencement of testing.

6.1.2 Discovery

Bluetooth is entirely responsible for devices discovery. The tests revealed that discovery did not work at every attempt. Discovery was successful in approximately 90 percent of tests conducted. The length of time consumed before discovery allowed a connection to be established ranged considerably. An average time of 30-90 seconds was calculated, with the quickest being under 10 seconds, and the longest being almost 3 minutes.



Talk Life: 3 hours (180 mins)	Standby: 195 hrs (8.1 days)

Table 6.1: Manufacturer Battery Specification

6.1.3 Range

The range of Bluetooth connectivity was tested by devices exchanging messages at defined distances. In the indoor test environment the maximum distance achieved was 12.5 metres and the minimum was 3 metres. It became apparent that if one user turned and moved more than a few metres very quickly the connection is much more likely to be broken than in a steady, controlled test. Outdoors the range is reduced considerably, with the Bluetooth standard range of 10 metres being difficult to achieve.

6.1.4 Battery

The manufacturer's specification quotes the battery times outlined in table 6.1 for the LiIon 850 mAh battery used in the Nokia 6600. With both phones fully charged, the application was altered to run an infinite loop of message exchange. The battery lasted less than 11 hours, almost half what is expected from continuous use in gaming applications. This shows that although Bluetooth is suited to low powered devices, continuous data transfer over RFCOMM is quite draining on the mobile phone's limited battery. A comparison of specification times, gaming estimation and the tested application is illustrated in figure 6.1.

6.1.5 Memory

Reduced memory was initially identified as a limitation to mobile phone based application development. This was not applicable due to the presence of a memory card in the Nokia 6600. The memory used by the application in continuous execution was measured. The application itself was 69.1kb. The tables 6.2 and 6.3 show the memory statistics from both phones at various stages of testing.

During testing a constant use of memory was observed, proportional to the messages exchanged. This reflects the persistent storage of messages on the device. This is illustrated by the graph in figure 6.2 compiled by the actual figures obtained.

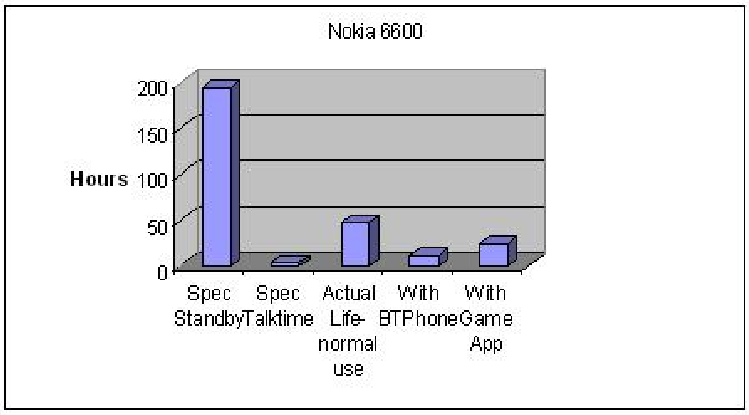
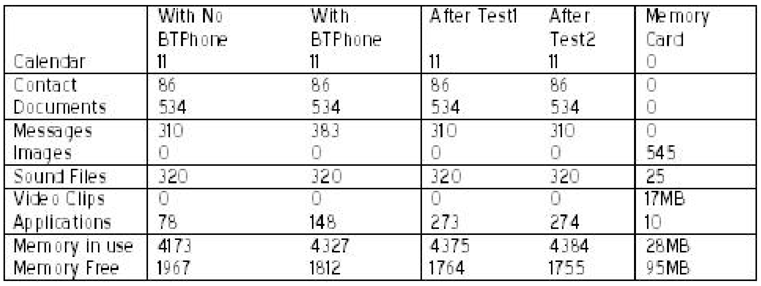
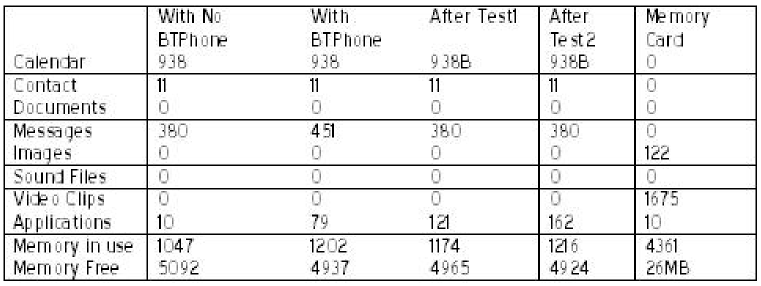


Figure 6.1: Comparison of Battery Consumption



	With No BTPhone	With BTPhone	After Test1	After Test2	Memory Card
Calendar	11	11	11	11	0
Contact	86	86	86	86	0
Documents	534	534	534	534	0
Messages	310	383	310	310	0
Images	0	0	0	0	545
Sound Files	320	320	320	320	25
Video Clips	0	0	0	0	17MB
Applications	78	148	273	274	10
Memory in use	4173	4327	4375	4384	28MB
Memory Free	1967	1812	1764	1755	95MB

Table 6.2: Memory in Device One



	With No BTPhone	With BTPhone	After Test1	After Test2	Memory Card
Calendar	938	938	938B	938B	0
Contact	11	11	11	11	0
Documents	0	0	0	0	0
Messages	380	451	380	380	0
Images	0	0	0	0	122
Sound Files	0	0	0	0	0
Video Clips	0	0	0	0	1675
Applications	10	79	121	162	10
Memory in use	1047	1202	1174	1216	4361
Memory Free	5092	4937	4965	4924	26MB

Table 6.3: Memory in Device Two

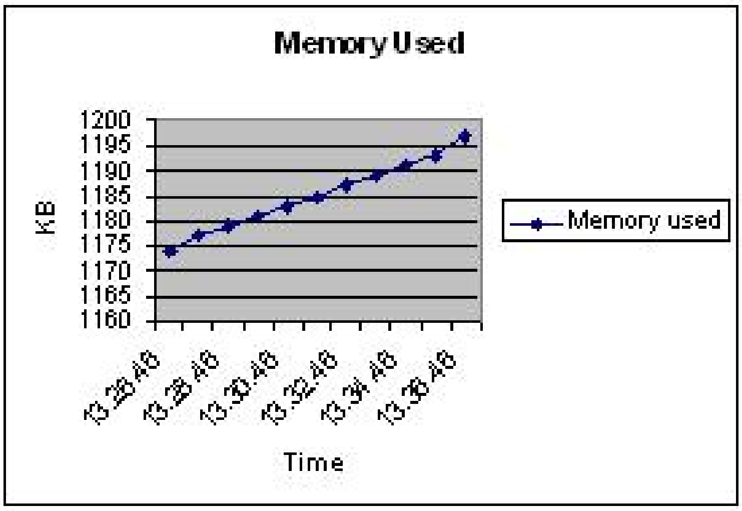


Figure 6.2: Memory Used

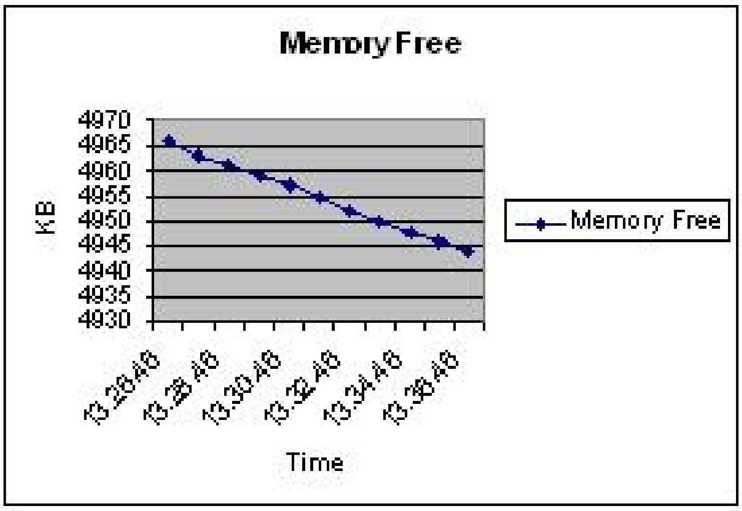


Figure 6.3: Memory Free

6.1.6 Performance

The evaluation of performance relates directly to the processing capacity of the mobile phone. This is obviously reduced, but is still quite powerful and adequate for most applications. The application was once again in an infinite loop of message exchange. Initially the screen updated showing the received message on average every 1-3 seconds. After 3 hours 1187 messages were displayed, which averages at 9 seconds per screen update. At this point, screen updates occurred approximately every 1 minute, although multiple messages may have been received within updates. Eventually screen updates took nearly 3 minutes and scrolling became a near impossibility. The application was stopped at this point. The number of messages stored on the phone was over 700 more than on display. This illustrates that the processor on the phone was capable of carrying out the underlying transport and Record Store functionality but the repainting of the canvas on the displayable screen object became a huge task as the number of messages to be printed grew.

Performance optimisation was not a primary objective of this dissertation. Refining the repaint() method in the display to show only a limited number of messages, and not the entire Record Store would significantly improve the perceived performance of the application.

Chapter 7

Conclusion

The final chapter documents the conclusions that can be drawn from the information presented in the exploration, design, implementation and evaluation of the context aware mobile phone based project undertaken in this dissertation. The primary aim was to investigate the area of context awareness on mobile phones with specific interest in Bluetooth applications. Finally, areas that could be considered for future work are identified.

Following the development of the mobile phone based application using Bluetooth, some general conclusions can be drawn. These share some similarities with conclusions of related work. The general conclusions drawn are the following:

7.0.7 Bluetooth

Bluetooth was found to be suitable for communication between mobile phones, although connection reliability and range cause doubt in its practical use. JABWT provides techniques to manage Bluetooth functionality, but has a sizable learning curve due to the number of objects involved and the technicalities of Bluetooth as a technology.

Bluetooth development id almost totally Linux based, and this was apparent at every attempt in every stage of development. From installation to low level access, Windows encountered Bluetooth issues which were addressed in multiple ways in Linux. Any future unrelated Bluetooth development would be conducted using Linux and

BlueZ, the most commonly used Bluetooth stack.

7.0.8 Mobile Phones

The first conclusion drawn regarding mobile phones is a direct consequence of the hardware challenges encountered. Mobile phones are not used to their potential by the majority of users. A growing percentage of users may occasionally use the embedded technologies, such as Bluetooth, for menial tasks. A very small percentage of mobile phones are used in application development. Manufacturers are aware if this fact, and it is evident from the 'Technical Reports' they published that aspects of the phone used in development are often not fully functional. There are a huge number of 'Known Issues' that inhibit comprehensive application development on mobile phones. The most significant example of this encountered in this dissertation is the inability of a Nokia 7610 to perform communication using a primary Bluetooth protocol. This confirms the conclusion that mobile phones may not fully conform to their advertised specification.

Interoperability is an area of difficulty when developing for mobile phones. The lack of interoperability between handsets made by different manufactures means that a universally compatible application becomes extremely difficult to achieve. The lack of a solution to allow interaction between Symbian applications and MIDlets greatly inhibits complex functionality involving the native information on the phone.

The conclusion that mobile phones lack sensor support confirms the findings of most research in the area of context awareness on mobile phones. The approach of adding external sensors is highly impractical, and so manual setting of context and the simulation of context capture is required. This severely restricts the ability of the mobile phone to become truly context aware.

Constraints imposed by the mobile phone were expected to inhibit application development. However, most of the restrictions, e.g. memory and storage, maximum jar size, were alleviated by the memory card within the handset. Screen size was not restricting, but this application was not graphically complex. The results of the evaluation phase present measurable results as to how much the application was affected by the physical constraints of the mobile phone. The phones processing power coped with the intensive testing without difficulty, with the only issue arising being the screen updates. The resource that was most adversely affected was the battery life.

7.1 Future Work

Due to time restrictions and hardware challenges encountered during implementation, some areas of the application were not investigated fully. Future work in these areas could extend the Bluetooth application functionality, and improve its performance. These are:

- Distributed issues: Issues including node failure, exception handling and network partitions were not investigated in great detail. By incorporating techniques to handle these areas the reliability and quality of service provided could be improved. The applications would be made more robust and fault tolerant.
- Incorporate Laptop: The inclusion of a Bluetooth enabled laptop or PC would add another dimension to the system created. Connectivity between the mobile phone and laptop using the serial port profile was established, but initial hardware difficulties led to the abandonment of the laptop as another Bluetooth node.
- Calendar Items: The ability to access native data from the phone would add a new source for the discovery of contextual information. The PIM API is available on some mobile phone models, and may be applicable for this behaviour.
- Context Based Decisions: User context is obtained and stored during the execution of the application, but the context is not used to its full potential. The application could be extended to use a rule based system to make decisions based on the context knowledge available. This would allow more intelligent choices. Filtering of messages could be performed based on a user's context e.g. age, availability. Nodes could also be eliminated from certain functions due to some contextual information known about them. This would be particularly useful for

prohibiting nodes that are on the periphery of the network from participating while they remain at such a distance. This could be seen as a form of predicting nodes likely to break connections and cause failure. This is a proactive approach to failure. Extending the application to be capable of context based decisions would increase its status as a context aware application.

- Performance Improvement: The evaluation revealed some performance issues when repainting the screen display. This area could be refined and perceived performance improved to the user. The code could also be refactored to increase efficiency and further improve performance.
- Security: The use of authentication and encryption in Bluetooth was not explored and could add safety precautions to avoid unauthorized users. The signing of MIDlets to create trusted applications was not undertaken either. The implementation of these security features could prevent bogus applications or users affecting the handset.

Bibliography

[1] Bluetooth transport for jxme. Jadabs Project, http://www.jadabs.de.

[2] H. Gellersen A. Schmidt, T. Stuhr. Context phonebook; extending mobile phone applications with context.

[3] Norman Adams Roy Want Bill Schilit. Context-aware computing applications.

[4] P J Brown, N Davies, M Smith, and P Steggles. Towards a better understanding of context and context-awareness. In H-W Gellerson, editor, *Handheld and ubiqitous computing*, number 1707 in Lecture Notes in Computer Science, pages 304–7. Springer, September 1999.

[5] J. Haller R. Kilian-Kehr C. Decker, S. Nguissi. Proximity as a security property in a mobile enterprise application context. 37th Hawaii International Conference of Systems Sciences, IEEE, 2004.

[6] M. Ficco V. Vecchio-S. Russo D. Cotroneo, F. Cornevilli. Implementing positioning services over an ubiquitous infrastrcture.

[7] J. Furukawa N. Moraveji-K. Reiger J. Shaffer D. Siewiorek, A. Smailagic. Sensay: A context-aware mobile phone.

[8] P. Dourish and V. Bellotti. Awareness and coordination in shared workspaces. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'92)*, pages 107–114, Toronto, Ontario, 1992. ACM Press.

[9] M. Vazirgianis G. polyzos-K. Norvag E. Valavanis, C. Ververidis. Mobishare: Sharing context-dependent data and services from mobile sources.

[10] SMS Explained. www.activexperts.com/asmssrvr/sms/smstech/.

[11] SMS Explained. www.activexperts.com/asmssrvr/sms/smstech/.

[12] Andreas Frei. Smart service archtecture for small devices.

[13] J. Hong S. Long-R. Kooper M. Pinkerton. G. Abowd, C. Atkeson. Cyberguide: A mobile context aware tour guide.

[14] Wikipedia GPRS. http://en.wikipedia.org/wiki/gprs.

[15] David Kotz Guanling Chen. A survey of context aware mobile computing research.

[16] Leslie Haddon. The social consequences of mobile telephony. *Ling Thrane (eds.)*, 2000.

[17] Carol Hamer. *J2ME Games with MIDP2*. Apress, 2004.

[18] Henk de Poot-David Langley. Henri ter ofte, Ingrid Mulder. I m mobile, where r u?

[19] http://jadabs.berlios.de/. Jadabs project.

[20] Infrared. http://www.dataacquisitionweb.com/interfaces/infrared.

[21] M.M. Lankhorst J. de Heer, A.J.H. Peddemors. Context-aware mobile business applications. In *Position paper of the first CoCoNet workshop, C̈ontext Aware Collaborative Environments for Next Generation Business Networks ̈*

, Zrich, Switzerland, 3-4 October 2002.

[22] Danial Kappeli. Jxta over bluetooth. Master's thesis, Swiss Federal Institute of Technology, Zurich, May 2003.

[23] CTO Rococo Software Karl McCabe. Time for a brew: Java and bluetooth.

[24] Navaneeth Krishnan. The jxta solution to p2p. *Javaworld*, October 2001.

[25] C. Bala Kumar, Paul J. Kline, and Timothy J. Thompson. *Bluetooth Application Programming with the Java APIs*. Morgan Kaufmann Publishers, 2003.

[26] Qing Li, Xiang Li, Jian Zhai, and Wenyin Liu. Mires: an information exchange system for mobile phones. In *SAC*, pages 1196–1200, 2004.

[27] D. Timmermann M. Handy, F. Golatowski. Lessons learned from developing a bluetooth multiplayer game.

[28] Sun Understanding J2ME Application Models. http://developers.sun.com/techtopics/mobility/midp/articles/models/.

[29] M.-E Mostafa. Mms - the modern wireless wolution for multimedia messaging. *The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2002.

[30] Nokia. www.nokia.com.

[31] Abhishek Pramod Patil. Performance of bluetooth technologies and their applications to location sensing. Master's thesis, Michigan State University, 2002.

[32] BlueCove Project. sourceforge.net/projects/bluecove/.

[33] JXTA-J2ME (JXME) Platform Project. jxme.jxta.org.

[34] P. Turner R. Cox, C. Newell. You, me and the otherness. *IADIS International Conference e-Society 2004*.

[35] Ferris Research. www.wi-fiplanet.com/tutorials/article.php/1577551.

[36] Peter Parnes Roland Parvainen. Supporting e-meetings on java capeble mobile phones.

[37] Sun Bluetooth Tutorial. http://developers.sun.com/techtopics/mobility/apis/articles/blueto

[38] Jason Yipin Ye. Atlantis:location based services with bluetooth. Master's thesis, 2005. Dept of Computer Science, Brown University.