TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

Pulchowk Campus

Final Report
on
Minor Project

Mobile Desktop

Submitted to
Department of Electronics and Computer Engineering

Submitted by
Anjan Nepal 061BCT502
rockanjan@gmail.com
9741033456
Manish Modi 061BCT524
manishmodi524@gmail.com
9841508410

25 February 2008

Acknowledgment

We would like to thank our supervisor Mr. Jayaram Timsina for his continuous support and suggestions without who this project would not have been a success. We are also grateful to our lecturers Mr. Bikash Shrestha and Mr. Dipen Chapagain for their constructive comments and criticisms during the project development. Thanks also to Jwalanta Shrestha and Shishir Jha for their guidance in configuring Bluetooth in Linux.

We are also greatly indebted to all our friends who helped us make this project successful. Last but not least, thanks to all who used the application and gave feed back on what things can be done to enhance the application.

Anjan Nepal (061/BCT/502)

Manish Modi (061/BCT/524)

Computer Engineering, 3rd year

Pulchowk Campus

List of Figures

Abbreviations

VNC - Virtual Network Computing

J2SE - Java 2 Standard Edition

J2ME - Java 2 Micro Edition

CLDC – Connected Limited Device Configuration

MIDP - Mobile Information Device Profile

API - Applications Programming Interface

PAN - Personal Area Networks

UUID - Universally Unique identifier

GUI - Graphical User Interface

GPRS - General Packet Radio Service

Table of Contents

Abstract

Mobile Desktop is an attempt to build a VNC like application for mobile phones to ease the remote desktop access through the mobile phones. It allows the control of a remote computer from mobile phones with maximum functions. It implements both Bluetooth and GPRS technologies for the connection mechanism. Bluetooth connection is costless but is limited to a small area, while GPRS is a costly world wide connection.

Mobiles are small and almost every time stays with you where ever you go. So, this application removes the overhead of carrying heavier laptops or searching some desktops around if you want to access your remote computer.

To use Mobile Desktop we run the server in the PC we want to access and a client application in mobile phone to connect to the PC.

A very common business application of Mobile Desktop is in remote system administration, where it is used to allow administrators to take control of employee machines to diagnose and fix problems from their mobile phones.

It can also be used to provide a flexible hot-desking and road-warrior environment by allowing employees to access their office desktop and server machines using their mobile phones.

Introduction

World is moving faster and accessories are becoming smaller and portable. The size of computer is greatly reduced over time from desktop to laptop. But still the desire is unfulfilled as mobiles are getting its popularity, we want the entire thing to fit in the mobiles and that doesn't exclude our desktop.

Keeping this in mind we have developed the Mobile Desktop which moves our desktop to the mobile, so that we can access the desktop in our workplace or wherever it may be using our mobile. The connection can be either Bluetooth or GPRS.

When it comes to network, security issues arise. So, to prevent the malicious users trying to enter the PC with the Mobile Desktop server running, authentication of the client if made by the server before giving access to it. Moreover, to allow only authorized users to run the server, authentication of the user starting of the server is also made. This information is kept in a file with the passwords MD5 encrypted, so the password will not be visible to the users in plain text.

We will be running the server application on the PC, and client application on the mobile phone, so that mobile can access and control the PC. Most important feature of Mobile Desktop is that it allows server to run in both Linux and Windows platform. And, the same client application in the mobile can access the server at any platform of operating system. The intriguing aspect of the project is that many clients can have simultaneous access to the server. So, Mobile Desktop can be very useful application to many.

Technologies

Bluetooth

Bluetooth is an industrial specification for wireless personal area networks (PANs). Bluetooth provides a way to connect and exchange information between devices such as mobile phones, laptops, personal computers, printers, digital cameras, and video game consoles over a secure, globally unlicensed short-range radio frequency. The Bluetooth specifications are developed and licensed by the Bluetooth Special Interest Group.

Uses

Bluetooth is a standard and communications protocol primarily designed for low power consumption, with a short range (power-class-dependent: 1 meter, 10 meters, 100 meters) based on low-cost transceiver microchips in each device.

Bluetooth enables these devices to communicate with each other when they are in range. The devices use a radio communications system, so they do not have to be in line of sight of each other, and can even be in other rooms, as long as the received transmission is powerful enough.

Class	Maximum Permitted Power mW (dBm)	Range (approximate)
Class 1	100 mW (20 dBm)	~100 meters
Class 2	2.5 mW (4 dBm)	~10 meters
Class 3	1 mW (0 dBm)	~1 meter

It has to be noted that in most cases the effective range of class 2 devices is extended if they connect to a class 1 transceiver, compared to pure class 2 network. This is accomplished by higher sensitivity and transmitter power of the Class 1 device. The higher transmitter power of Class 1 device allows higher power to be received by the Class 2 device. Furthermore, higher sensitivity of Class 1 device allows reception of much lower transmitted power of the Class 2 devices. Thus, allowing operation of Class 2 devices at much higher distances.

Version	**Data Rate**
Version 1.2	1 Mbit/s
Version 2.0 + EDR	3 Mbit/s
WiMedia Alliance (proposed)	53 - 480 Mbit/s

GPRS

General Packet Radio Service (GPRS) is a Mobile Data Service available to users of Global System for Mobile Communications (GSM) and IS-136 mobile phones. It provides data rates from 56 up to 114 kbps.

GPRS can be used for services such as Wireless Application Protocol (WAP) access, Short Message Service (SMS), Multimedia Messaging Service (MMS), and for Internet communication services such as email and World Wide Web access.

2G cellular systems combined with GPRS is often described as "2.5G", that is, a technology between the second (2G) and third (3G) generations of mobile telephony. It provides moderate speed data transfer, by using unused Time division multiple access (TDMA) channels in, for example, the GSM system. Originally there was some thought to extend GPRS to cover other standards, but instead those networks are being converted to use the GSM standard, so that GSM is the only kind of network where GPRS is in use. GPRS is integrated into GSM Release 97 and newer releases. It was originally standardized by European Telecommunications Standards Institute (ETSI), but now by the 3rd Generation Partnership Project (3GPP).

J2ME

In computing, the Java Platform, Micro Edition or Java ME (previously known as Java 2 Platform, Micro Edition or J2ME) is a specification of a subset of the Java platform aimed at providing a certified collection of Java APIs for the development of software for small, resource-constrained devices such as cell phones, PDAs and set-top boxes.

Java ME was designed by Sun Microsystems and is a replacement for a similar technology, PersonalJava. Originally developed under the Java Community Process as JSR 68, the different flavors of Java ME have evolved in separate JSRs. Sun provides a reference implementation of the specification, but has tended not to provide free binary implementations of its Java ME runtime environment for mobile devices, rather relying on third parties to provide their own.

Usage

Java ME has become a popular option for creating games for cell phones, as they can be emulated on a PC during the development stage and easily uploaded to phones. This contrasts with the difficulty of developing, testing, and loading games for other special gaming platforms such as those made by Nintendo, Sony, Microsoft, and others, as expensive system-specific hardware and kits are required.

Java ME devices implement a profile. The most common of these are the

Mobile Information Device Profile aimed at mobile devices, such as cell phones, and the Personal Profile aimed at consumer products and embedded devices like Set-top boxes and PDAs.

Profiles are subsets of *configuration*s, of which there are currently two: the Connected Limited Device Configuration and the Connected Device Configuration.

Connected Limited Device Configuration

The Connected Limited Device Configuration (CLDC) contains a strict subset of the Java class libraries, and is the minimal needed for a Java virtual machine to operate. CLDC is basically used to classify myriad devices into a fixed configuration.

A configuration provides the most basic set of libraries and virtual-machine features that must be present in each implementation of a J2ME environment. When coupled with one or more profiles, the Connected Limited Device Configuration gives developers a solid Java platform for creating applications for consumer and embedded devices.

Mobile Information Device Profile

Designed for cell phones, the Mobile Information Device Profile boasts GUI API, and MIDP 2.0 includes a basic 2D gaming API. Applications written for this profile are called MIDlets. Almost all new cell phones come with a MIDP implementation, and it is now the de facto standard for downloadable cell phone games. However, many cell phones can run only those MIDlets that have been approved by the carrier, especially in North America.

JSR 271: Mobile Information Device Profile 3 will specify the 3rd generation Mobile Information Device Profile (MIDP3), expanding upon the functionality in all areas as well as improving interoperability across devices. A key design goal of MIDP3 will be backward compatibility with MIDP2 content.

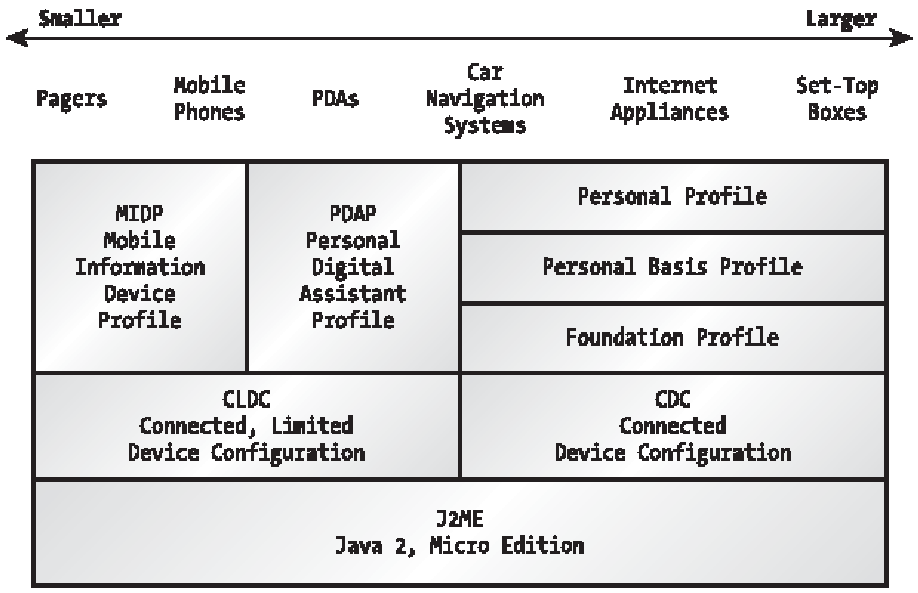

Smaller
Larger
Pagers
Mobile Phones
PDAs
Car Navigation Systems
Internet Appliances
Set-Top Boxes
MIDP Mobile Information Device Profile
PDAP Personal Digital Assistant Profile
Personal Profile
Personal Basis Profile
Foundation Profile
CLDC Connected, Limited Device Configuration
CDC Connected Device Configuration
J2ME Java 2, Micro Edition

Fig 1: Common J2ME profiles and configurations

Bluetooth Stack

A Bluetooth stack refers to an implementation of the Bluetooth protocol stack.

Bluetooth stacks can be roughly divided into two:

i. General-purpose implementations that are written with emphasis on feature-richness and flexibility, usually for desktop computers. Support for additional Bluetooth profiles can typically be added through drivers.

ii. Embedded system implementations intended for use in devices where resources are limited and demands are lower, such as Bluetooth peripheral devices.

Generally, only a single stack can be used at any time: switching usually requires uninstalling the current stack, although a trace of previous stacks remains in the registry. However, there are some cases where two stacks can used on the same computer, each using their own separate Bluetooth radio hardware.

BlueZ

BlueZ is the official Bluetooth stack for Linux. Its goal is to make an implementation of the Bluetooth wireless standards specifications for Linux. As of 2006, the BlueZ stack supports all core Bluetooth protocols and layers. It was initially developed by Qualcomm, and is available for Linux kernel versions 2.4.6 and up.

In addition to the basic stack, the Bluez-utils and Bluez-firmware packages contain low level utilities such as dfutool which can interrogate the Bluetooth adapter chipset to determine whether its firmware can be upgraded.

External Libraries Used

AvetanaBT

AvetanaBluetooth is a Java/JNI-Implementation of JSR-82 for Linux and J2SE. AvetanaBluetooth allows programmers to easily use and offer Bluetooth services.

Official Documentation can be found at

http://sourceforge.net/projects/avetanabt

BlueCove

BlueCove is a Java library for Bluetooth (JSR-82 implementation) that currently interfaces with the Mac OS X, WIDCOMM, BlueSoleil and

Microsoft Bluetooth stack found in Windows XP SP2 or Windows Vista and WIDCOMM and Microsoft Bluetooth stack on Windows Mobile.

BlueCove can be used in Java Standard Edition (J2SE) 1.1 or newer.

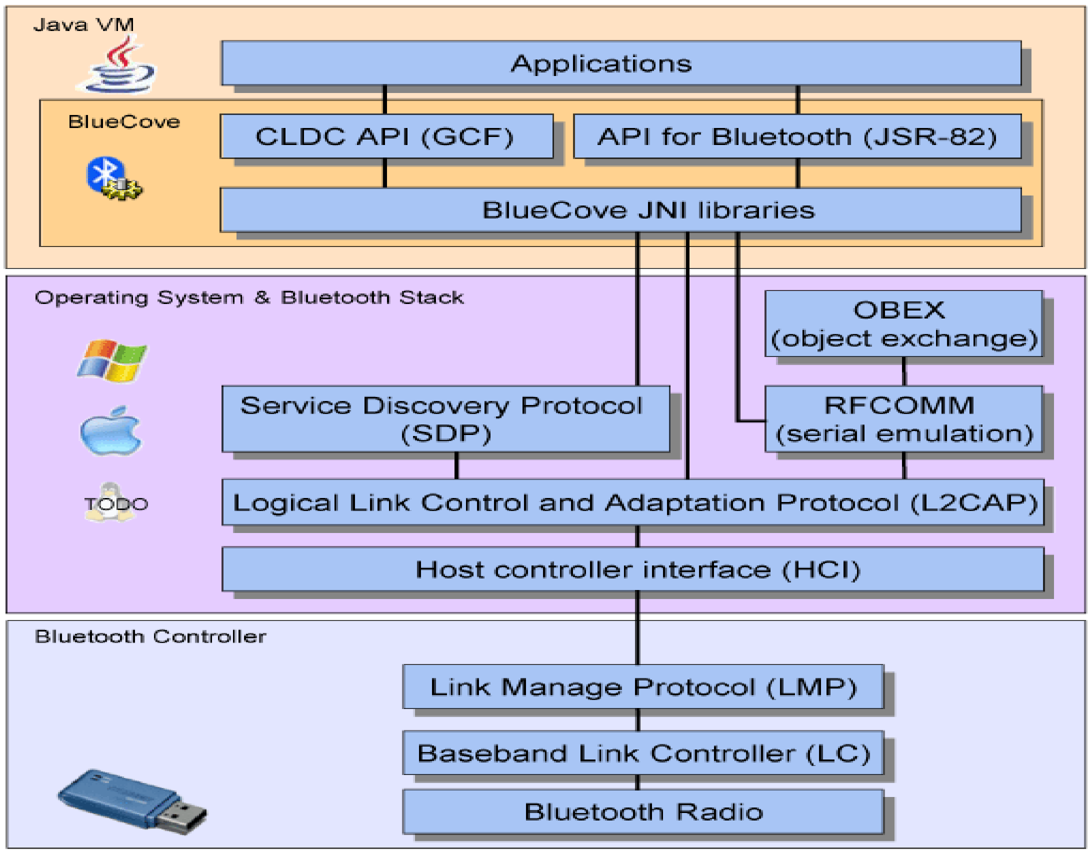

Java VM
Applications
BlueCove
CLDC API (GCF)
API for Bluetooth (JSR-82)
BlueCove JNI libraries
Operating System & Bluetooth Stack
OBEX (object exchange)
Service Discovery Protocol (SDP)
RFCOMM (serial emulation)
TODO
Logical Link Control and Adaptation Protocol (L2CAP)
Host controller interface (HCI)
Bluetooth Controller
Link Manage Protocol (LMP)
Baseband Link Controller (LC)
Bluetooth Radio

Fig 2: Bluetooth Implementation of Bluecove

Infrastructure

This section provides the description of infrastructure we used for the development of Mobile Desktop.

Operating System

We used both Linux and Windows for the development of our application. The platform independence was main target of our project. We used Java as our programming tool, which provides platform independence. But for the Bluetooth connection, there were no APIs in J2SE, so we used external APIs which were OS dependent. So we used different Bluetooth library for Linux & Windows. AvetanaBT was used in Linux and Bluecove was used in Windows XP SP2 as APIs for Bluetooth support in J2SE.

Java Enabled Mobile Phones

We used different Java enabled Mobile phones for the testing of our application. In order to provide more flexibility to our application we used different mobile phones to eliminate vendor specific application. We successfully tested our projects on Sony Ericsson 810i, Nokia 3110c, Nokia 6255i and Nokia 6131.

For our application Mobile phone must have MIDP 2.0 and CLDC 1.0 or higher and also should have a serial port profile (in case of Bluetooth connection).

Though we have tried to provide backward compatibility to MIDP 1.0, certain features are eliminated and it is yet to be thoroughly tested.

PC with Bluetooth Extension

We need Desktop/Laptop computers with Bluetooth extensions to establish connection between Server & Client.

Bluetooth connection may be obtained either with Laptop with Bluetooth built inside it or can use Bluetooth dongle.

Bluetooth dongles are the simplest way of adding Bluetooth functionality to any PC.

During the development of this project, the Bluetooth connectivity was acquired using a USB Bluetooth Dongle. The application has also been tested with success in built-in Bluetooth of a Laptop.

Platform and Development Environment

Mobile Desktop application has basically server running on the PC and client running on Mobile Phones. The platforms and development environments used during the project development are discussed here.

Server

J2SE (core Java) has been used for the development of server. Netbeans 5.5 and 6.0 beta were been used as an IDE for the coding, compiling & debugging. Windows XP SP2 was used for the server application coding for this operating system. Bluecove needs Windows XP SP2 for it's APIs to be accessed. And, Kubuntu and Ubuntu were used for the server application coding for Linux.

Client

J2ME has been used for the development of the client (mobile). Netbeans 5.5 with mobility pack and Netbeans 6.0 beta were used as an IDE for coding, compiling, emulating and debugging the client side application.

GUI

The GUI has been coded in Java Standard Edition (J2SE). The GUI was developed using Java's Swing Package. The tool used for this purpose was Netbeans Swing Utility.

Methodology

Full fledged working Mobile Desktop application has following functional components

i. **The GUI**

ii. **The Server**

iii. **The Client**

The GUI

Overview

GUI is Java based program that acts as an interface to allow various administrative works on the Server. It is used to create the new user, update it and remove it. Basically it displays the various connection and processing logs.

GUI performs following task

- It opens the graphical interface for the program.
- It performs start and stop server operations.
- It is used to accept and disconnect clients.
- It shows actively connected clients.
- It displays the connection and processing logs.
- It creates new user for client.
- It updates and removes the existing users.

Implementation

The GUI is a desktop application which is seen after the jar file of the Mobile Desktop Server is executed.

While running, the program does the following jobs.

- Provides the options to start and stop server.
- Shows number of clients connected.
- Shows the log of clients.
- Does administration tasks i.e. create, remove, update user.

The Server

Overview

Server is program that runs in the background of Mobile Desktop Server application; it establishes the connection with different clients and serves their requests.

Basic functions performed by the Server listed below

- Server program runs on the PC which has to be remotely accessed.
- It is secured with passwords.
- It handles the request from client.
- It authorizes the client and gives access to them.
- It can handle multiple client simultaneously.

- Provides more flexibility as it can run on both Linux and Windows Platform.

Design

Server comprised of following functional components which are integrated to provide overall server functionality

i. Authentication Module

ii. Connection Engine

iii. Request Handling Engine

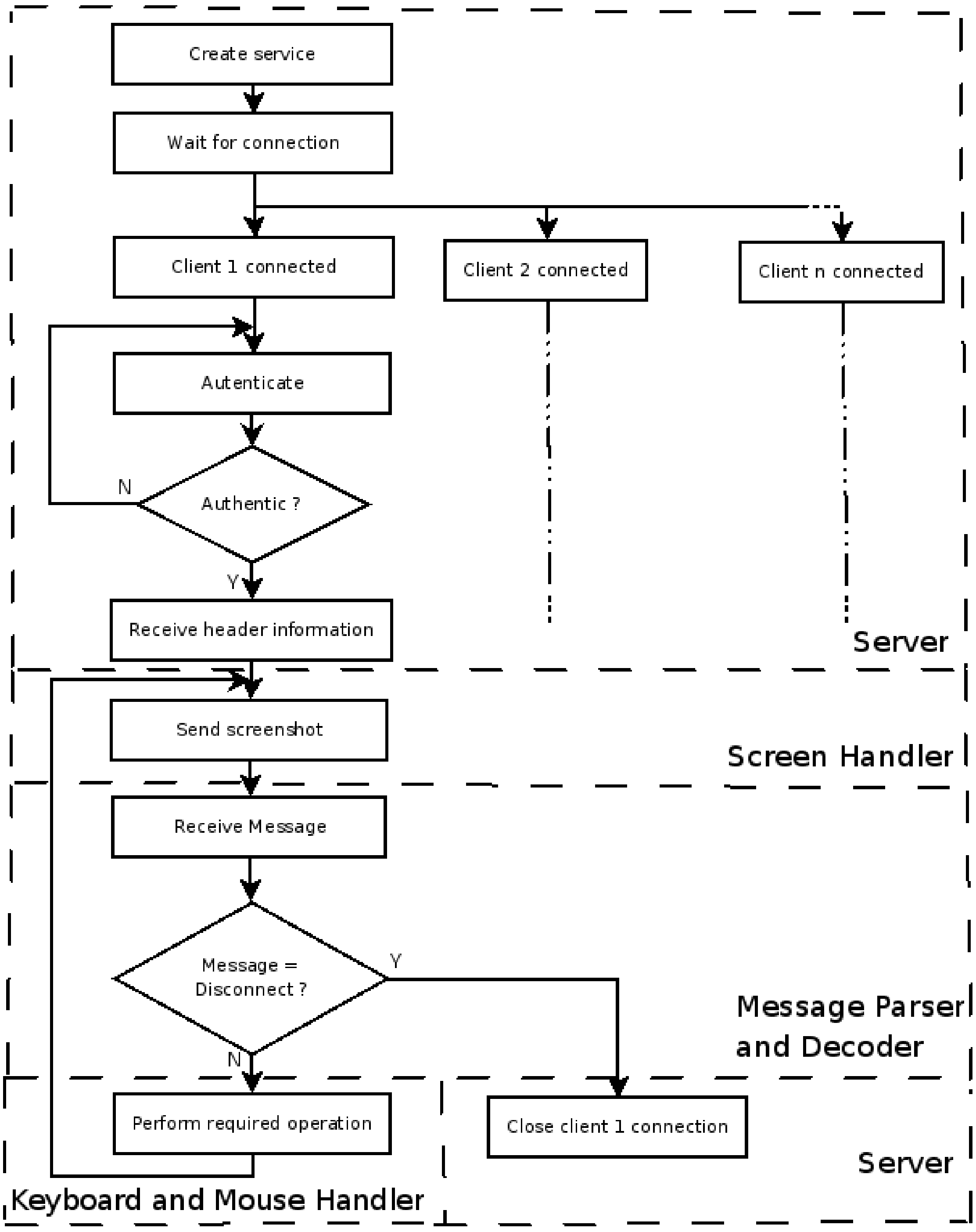


Fig 3: Flow of the Mobile Desktop Server

Authentication module

It performs authentication for server as well as clients. It provides flexibility with various operations. Basically it provides the security to the Mobile Desktop Application. It uses MD5 encryption for storing the login information.

Operations that authentication module provides are listed below

- Add User
- Remove User
- Update User
- Authenticate User

Connection Engine

It handles the connection between the server and client. It remains in continuous looping state for receiving the connection requests from clients and establish the connection to them.

We have used Bluetooth and GPRS for the connection between Server and Client.

Basic Implementation of Connection using Bluetooth is listed below.

Connection using Bluetooth

- Initialize Bluetooth service on Remote PC.

- Create service for the program with UUID.

- Accept the request from the client in its vicinity with the given UUID.

To provide better flexibility to our application we have also used GPRS connection to for Server-Client connection. The application has been successfully tested by using GPRS connection. But, due to slow GPRS connection provided by the mobile service provider (Mero Mobile) the results were very slow. So, a faster GPRS connection can show the full potential of the application like with the Bluetooth connection.

Basic Implementation of Connection using GPRS is listed below.

Connection using GPRS

- GPRS connection through IP address and port

- Server running remotely must have public IP

- It receives request from client.

- If the client is authentic then server gives access to it.

- Connection uses TCP/IP connection i.e. no data loss

Request Handling Engine

It is the core engine of Mobile Desktop which performs all the operations after the connection till the client is disconnected. It gets requests from client, synthesizes it and sends the output to client.

Request Handling Engines has following functional components

i. Message Parser , Decoder and Executer

ii. Message Information Header

iii. Mouse Handler

iv. Keyboard Handler

v. Screen Handler

Message Parser, Decoder and Executer

It parses the message obtained from client to the header and information. The information obtained is decoded and then subjected to execution.

Message Information Header

It contains the header information for every request and their corresponding action.

Mouse Handler

It handles the mouse event requested by the client. It performs all the system mouse event.

Keyboard Handler

It handles the keyboard event requested by the client. It performs all the system keyboard event.

Screen Handler

It handles the display related event requested by the client. It performs all the action on the display. It keeps track of the output after the client operations performed. It gives the screen in the client's view-port as is the result expected. An example is that, if the client is in the fit-screen mode the server sends the whole screen to the client after scaling to the view-port size of the client.

The directory structure might look like: (+ represents folder, - represents file)

+ mobiledesktopserver

  - Server.java
  - ServerHandler.java
  - ScreenHandler.java
  - MouseHandler.java
  - KeyboardHandler.java
  - Information.java
  - Authorize.java

Implementation

The Server is a Linux/Windows based command line application which is invoked by the following command.

```
java -jar MobileDesktopServer.jar
```

While running, the program does the following jobs.

- In case of Bluetooth, initializes the Bluetooth Serial Port service and starts the service and waits for a client to connect. In case of GPRS connection it starts the SocketServer connection and waits for a client to connect.
- Accepts the connection.
- Receives the Header message from the mobile phone over Bluetooth. Header message contains the device width and height.
- Stores the device information.
- Listens to client request.
- Performs the requested action.
- Sends the new screen shot after the action is performed to the client.

Client

Overview

Client is the mobile application which searches the server and connects to it. Once the server is found, the authentication information is sent to the server and if the authentication is valid, the server does the requests of the client.

The basic functions of client are

- It runs on the mobile devices with CLDC 1.0 & MIDP 2.0 or higher.
- In case of Bluetooth, it searches the service with given UUID of the Mobile Desktop service in vicinity i.e. Server. In case of GPRS, it tries to connect to the server using the IP and port address the user provides.
- It provides choice for the full screen mode or normal mode.
- It accesses the server with valid username and password.
- It can discover and connect to the server even if there are other Bluetooth enabled mobiles around.
- It runs independently even if there are other mobiles devices connected to same server.

Design

Client contains the following components.

i. Connection Engine

ii. Preferences & Authentication Module

iii. User Interface

iv. Request Handling Engine

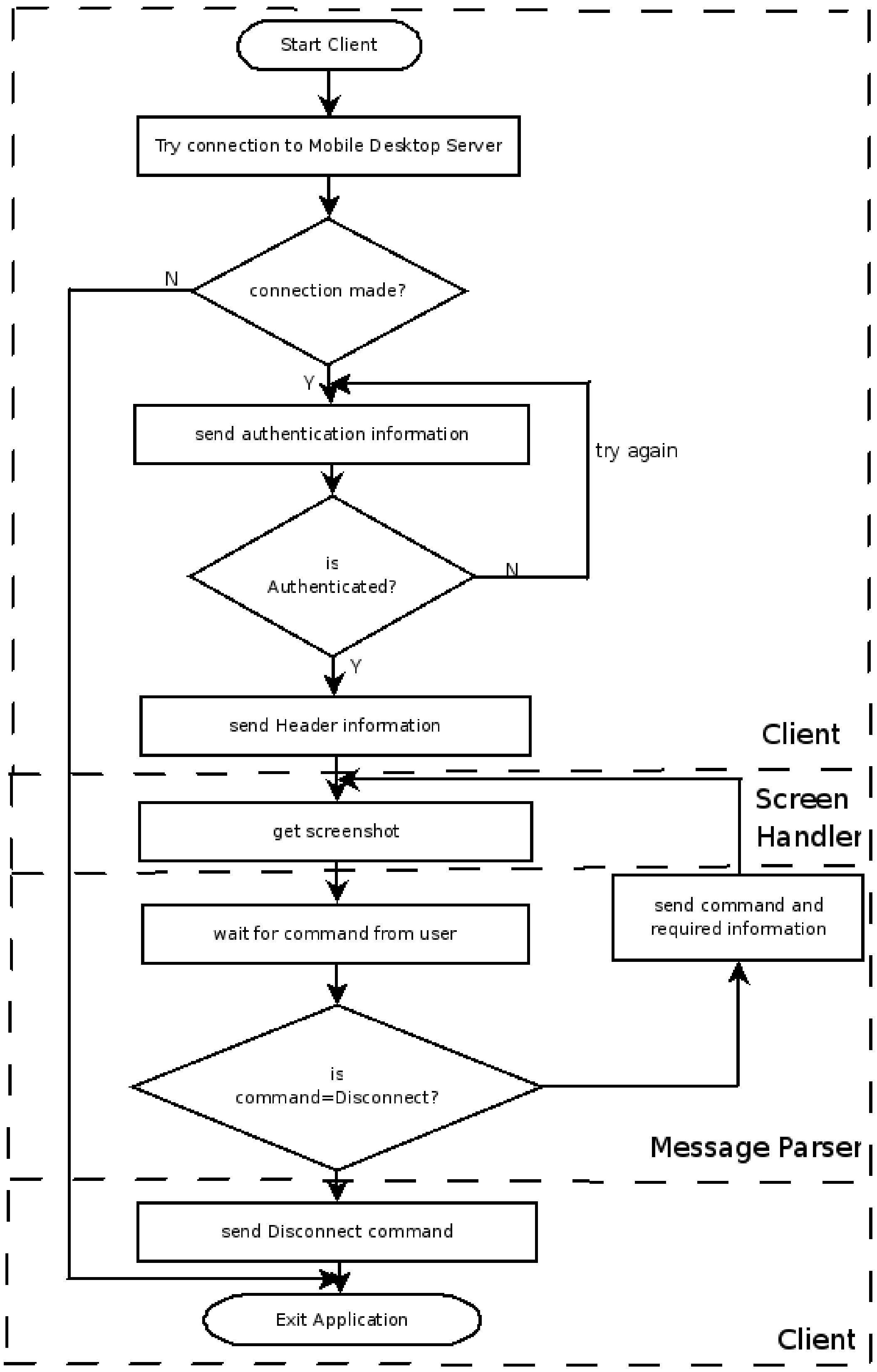


Fig 4: Flow of Mobile Desktop Client

Connection Engine

It handles the connection between the client and server. It can connect to the server through any of the given connection options.

i. Bluetooth

ii. GPRS

We have used Bluetooth for the connection between Server and Client.

Basic Implementation of Connection using Bluetooth is listed below.

Connection using Bluetooth

- Search the service with given UUID of Mobile Desktop Server.
- If Server is found select and connect to the service otherwise exit the application.

As we have already mentioned still Bluetooth is good option for connection we need to implement the GPRS for remote connection through the internet so we have created the GPRS connection option to the client. We have successfully tested the GPRS connection in a mobile but the results were very clumsy due to low GPRS connection speed.

Basic Implementation of Connection using GPRS is listed below.

Connection using GPRS

- GPRS connection through IP address and port

- Client running remotely must try to access the server with public IP in the specified port.
- After connection, sends the authentication information to the server.
- If the client is authentic then server gives access to it.
- Connection uses TCP/IP connection i.e. no data loss

Preferences & Authentication

We provide the client with the choice to connection mechanism i.e. either through Bluetooth or GPRS. Besides that client can even have options to select the full screen mode or normal mode. The user name and password entered in the authentication form and the IP and port address in the GPRS connection form from the last access of the application is saved for the next time. This reduces the overhead of typing the information over and over again if they are same each time

Preferences module consists of following part

Connection Option Form

It is user interface which enables the user to choose one of the options for connection, either Bluetooth or GPRS connection.

Full screen Option Form

It is the user interface that provides the user with option to choose between full screen and normal view.

IP and Port Selection Form

It is interface for users which prompts user to enter the IP address and port number of the Server in GPRS connection.

We have authentication engine in the Server which verifies the valid user. So we have provided the client with login form to validate its authenticity.

Authentication implements the following module

Login Form

It is the interface which prompts the user to enter the valid user name and password.

User Interface

It is the interface to the user which enables the user to view the screen of the Remote PC and perform various operations. It enables to the user to browse through the screen of Server with various options. Options are powered by the various key combination to produce desire effect. All the keys and key combinations are mentioned in the appendix.

User Interface implements the following module to accomplish its tasks

Screen Handler

It provides user with various options to perform desired task. It allows user to move the mouse around the screen of mobile and type various text.

Request Handling Engine

Similar to the Server it is the also the Clients core engine which perform all the operations after the connection till the client is disconnected. It gets requests from user, synthesizes it and sends it to the Server and produce the output.

Request Handling Engines has following functional components

i. Message Information Header

ii. Keyboard Form

iii. Request Send & Get module

Message Information Header

It contains the header information for every request and their corresponding action.

Keyboard Form

It provides the interface for the user to enter their text and send it to the server.

Request Send & Get module

It handles the request between the server and client i.e. sends the request from the client to server & receives the output from the Server. E.g. Change the view port of the screen.

The directory structure looks like: (+ represents folder, - represents file)

+ mobiledesktopclient

    - Client.java

- ConnectionOptionForm.java
- FullscreenOptionForm.java
- GPRSForm.java
- AuthenticationForm.java
- ScreenHandler.java
- KeyboardForm.java
- Information.java
- Preferences.java

Implementation

The Client is a mobile application which is invoked by the clicking on their MobileDesktopClient.jad (java application descriptor file) or MobileDesktopClient.jar file in the mobile. Some mobiles launch the application just after clicking the jad or jar file. But some require installation before launching the application.

While running, the program does the following jobs.

- Prompt for screen view options, either full screen or normal mode.
- Prompts for connection option, either Bluetooth or GPRS
- If the option is GPRS, the IP and port address is asked in the GPRS form
- Connects to the server.

- Prompts for user authentication.

- After authentication by the server, sends device information to the server.

- Waits for user action.

- Creates the header and message for the user action.

- Sends the header and message to server.

- Receive the resultant screenshot from the Server.

Data Format

Client requests the Server to perform an action by sending the message to the Server and receives the screenshot from the Server. It creates the message and header according to a common static integers defined in Information.java file in both server and client side. We basically divided the message into two parts.

i. Header

ii. Information

Header

It is the command from the client to the server. Simply, it is an integer telling the server what is coming next.

Information

After the header, additional information might be needed for the task to be done by the server.

For example, if the keyboard text is to be sent to the server, the integer representing the command to type is sent to the server first. Then the server waits for the string. After client sends its string to the server, the server types the text and sends back the screenshot of the result.

Security Aspects

Security is important aspect of any application and Mobile Desktop application is not an exception. Since Mobile Desktop uses network and remote access of the Server, it is very important to prevent unauthorized access to the Server. In order to prevent unauthorized and malicious user to access the Server we have implemented the login system to allow only valid user to access the Server.

Once a user has login, he can look at the login information file and see other users account information. So, in Server side MD5 encryption has been used to store user and password information.

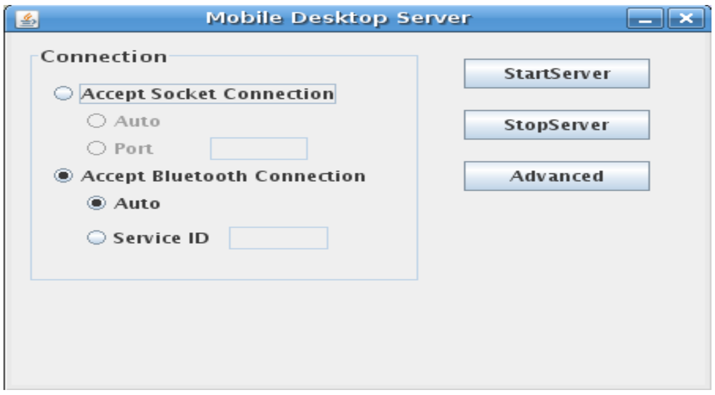
Usage Procedure

To set up Mobile Desktop, the following procedures have to be followed.

Installation

A client application is to be installed on Java enabled cell phone(MIDP 2.0 & CLDC 1.0). Similarly Server and front end application has to be installed on the PC.

Operation

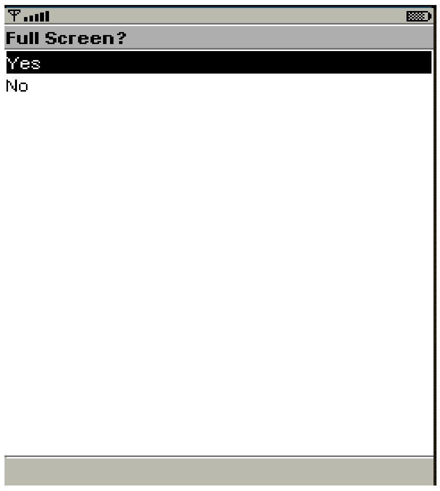


Connection

- Accept Socket Connection
  - Auto
  - Port
- [✓ Accept Bluetooth Connection]
  - [✓ Auto]
  - Service ID

Fig 5: Server GUI

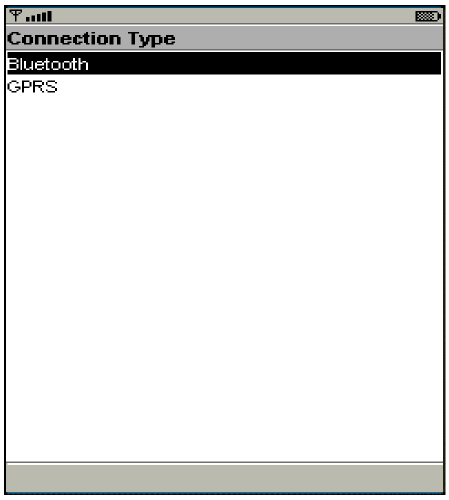
- ➢ The server is started in the PC and it waits for the clients to connect.
- ➢ The client application is started in the mobile
- ➢ The option for either full screen or the normal mode is prompted to the user.
- ➢ The connection preference option form prompting to enter the type of connection the user wants (Bluetooth or GPRS) appears.


Full Screen?

Yes

No

Fig 6: Full Screen Form

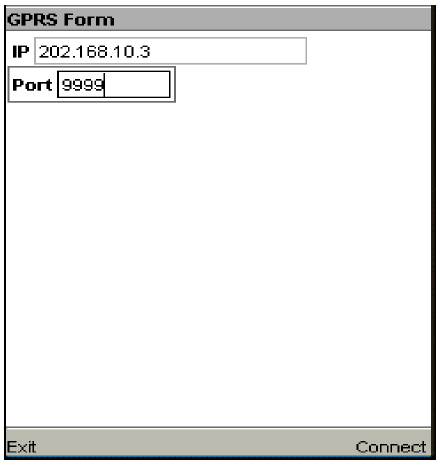

Connection Type

Bluetooth

GPRS

Fig 7: Connection Option Form

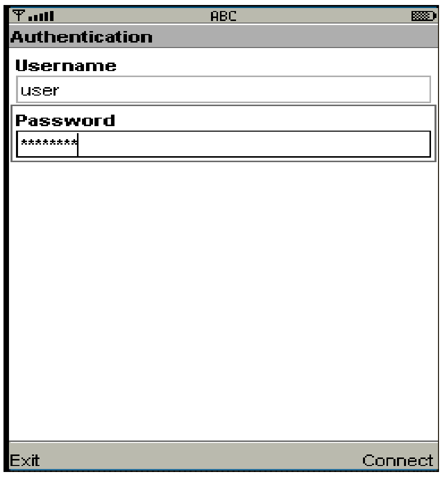
- The IP and port address of the server is prompted if user wants to make GPRS connection.
- The authentication form is displayed prompting the user to enter the username and password.


GPRS Form

IP 202.168.10.3

Port 9999

Fig 8: GPRS form


Authentication

Username: user

Password: ********

Fig 9: Authentication form

- If the authentication is valid, the first screenshot is received from the server. Requests from the user are constantly sent to the server and new screenshots are received after server performs the actions.

Limitations

Every application has limitations and so does Mobile Desktop. These are the main limitations seen so far.

- Though MIDP tries to make the things simpler by giving an API supported for all devices, the things get changed from mobile to mobile. One thing that our application is affected of is the button configurations. The button configurations can differ greatly from vendor and models. Though we have tried our best to make the buttons configuration same in all the mobiles we've tested and we are not sure that it works perfectly as expected in other mobiles.
- For Bluetooth connection, the application only supports mobiles with Serial Port Profile. Serial Port Profile is the serial port emulation of the Bluetooth enabled mobiles for the communication.

Further Works

These are few works which can be done in the future for enhancing the application.

- Presently the project can be implemented only in cell phones with MIDP 2.0 and CLDC 1.0 or higher. Hence among important works that needs to be done to make this project more portable and acceptable to a wider audience, portability to other devices that support MIDP 1.0 needs to be done.
- The limitation seen due to the Serial Port Profile has to be removed.
- Another important future enhancement required is OBEX file transfer from the server to the client, which will provide versatility to this application.

Conclusion

The project on Mobile Desktop was successfully completed. During the project development we became familiar with various tools of software development. Similarly, various technologies like Bluetooth and GPRS were used in programming and research in those areas gave a bigger picture of how they work. The project was also deliberately chosen as an objective of learning both J2ME and J2SE. Various challenges were encountered and were solved gradually. At first, the connection from the GPRS seemed hopeless, which was our proposed medium of connection, so we switched to Bluetooth connection. After it was successful with the Bluetooth connection, the GPRS connection programming was eased as well.

The mobile (client) should have CLDC 1.0 or higher and MIDP 2.0 or higher for running this application. A version of project for MIDP 1.0 is also available but has not been tested yet. The limitations seen this far will be removed and further enhancement will be added in the future versions.

The project can be found at http://www.sourceforge.net/projects/mobiledesktop

Bibliography

Books

1. Dietel & Dietel, How to program JAVA, Prentice Hall India
2. Jonathank Knudsen & Sing Li, Beginning J2ME from Novice to Professional, Apress

Links

3. Bluetooth
    - ➢ http://www.bluetooth.org
    - ➢ http://www.en.wikipedia.org/wiki/bluetooth
    - ➢ http://www.sourceforge.net/projects/avetanabt
    - ➢ http://www.bluecove.org/
    - ➢ http://www.bluez.org
4. Forum Nokia, http://forum.nokia.com
5. Forum Java, http://forum.java.sun.com/index.jspa

Appendix

Keys and their corresponding functions in the client side.

Non Fit-screen mode

Key	Function
Numpad 0	Switch to fit-screen mode
Numpad 1	Refresh
Numpad 2	Move Screen Up
Numpad 3	Right Click
Numpad 4	Move screen Left
Numpad 5	Keyboard input
Numpad 6	Move screen Right
Numpad 7	Press enter key
Numpad 8	Move screen down
Numpad 9	Press delete key
*	Zoom out
#	Zoom in
Arrow key up	Move mouse Cursor up
Arrow key down	Move mouse Cursor down
Arrow key left	Move mouse Cursor left
Arrow key right	Move mouse Cursor right
Command button 1	Left mouse click
Command button 2	Double click
Command button 3	Disconnect

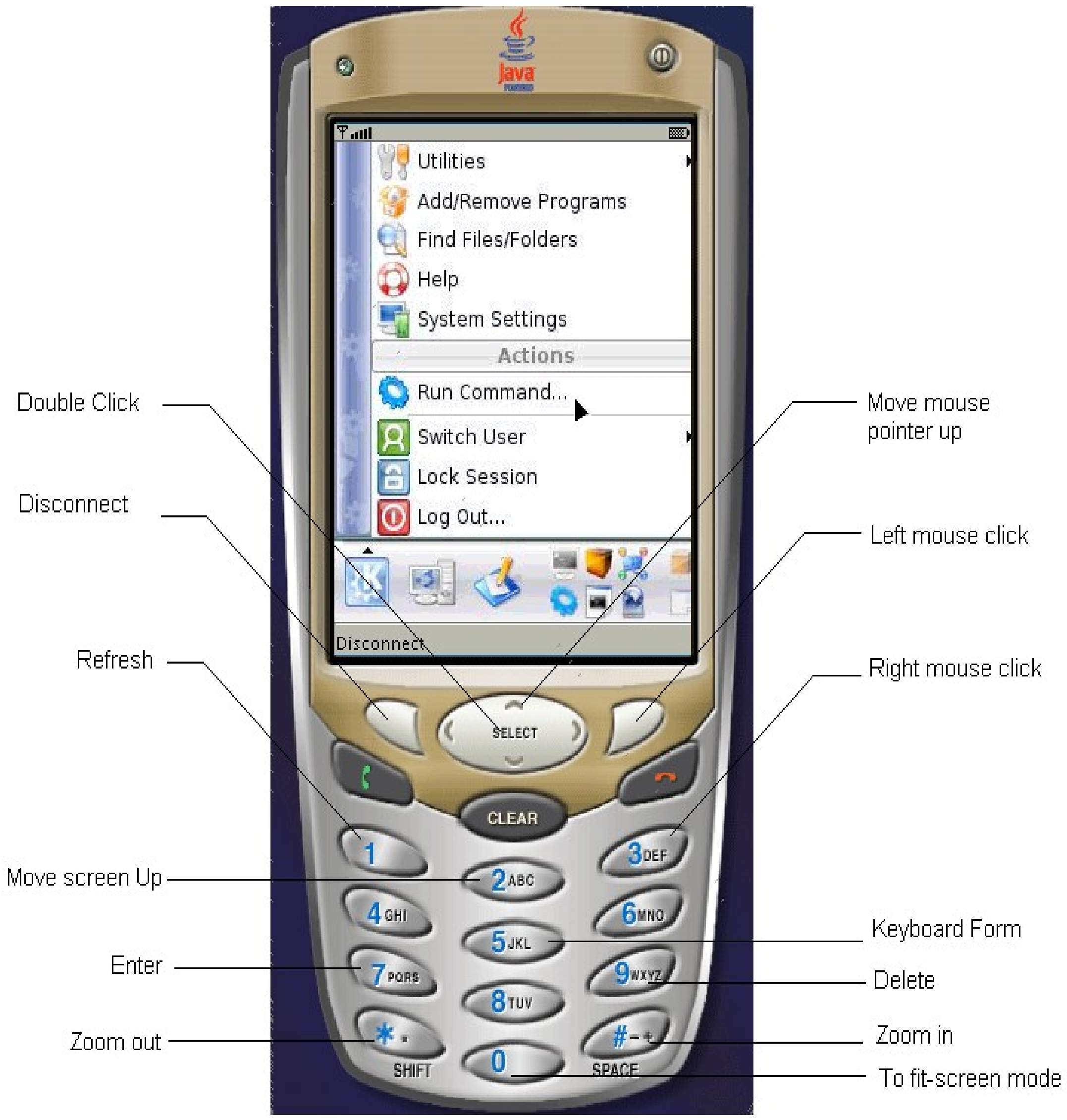


Fig 10: Keys and their function in Normal mode

Fit-screen mode (Specially designed for presentation)

Key	Function
Numpad 0	Switch to normal mode
Numpad 1	Refresh
Numpad 2	Press Up directional key
Numpad 3	Right Click
Numpad 4	Press Left directional key
Numpad 5	Press F5 button in the server side
Numpad 6	Press Right directional key
Numpad 7	Press enter key
Numpad 8	Press Down directional key
Numpad 9	Press delete key
*	Press escape key
#	Press backspace key
Arrow key up	Press Up directional key
Arrow key down	Press Down directional key
Arrow key left	Press Left directional key
Arrow key right	Press Right directional key
Command button 1	Left mouse click
Command button 2	Double click
Command button 3	Disconnect

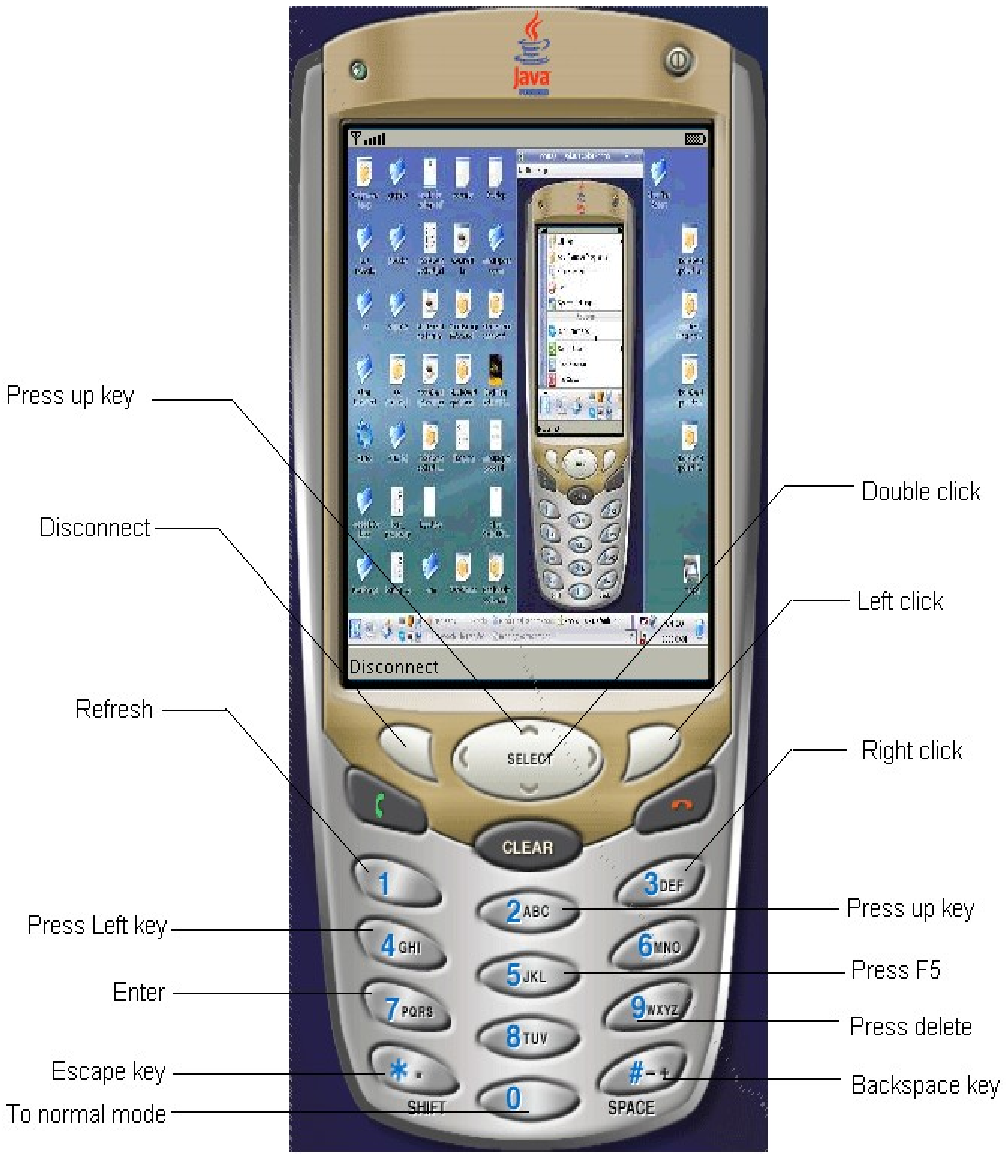


Fig 11: Keys and their function in Fit-screen mode