Implementing Methodologies for Achieving a Communication Channel between a Mobile Phone and a Remote Computer

Manasi Yerunkar,
Student
Department of Computer Science
MPSTME, NMIMS
Mumbai, India.

Amala Rangnekar
Student
Department of Computer Science
MPSTME, NMIMS
Mumbai, India.

Alpa Reshamwala
Assistant Professor
Department of Computer Science
MPSTME, NMIMS
Mumbai, India

Abstract: Ubiquitous computing is a post-desktop model of human-computer interaction in which information processing has been thoroughly integrated into everyday objects and activities. It enables a person located in any part of the network to control a remote PC and perform basic operations like running applications, downloading files etc. In this paper we have explained a Java-based application of Ubiquitous Computing for Computer Access through mobile phone. Also, we have explained two methods for achieving a communication channel between a mobile phone and a remote computer. The first method is using the hardware components like microprocessor and SIM card but we encountered some disadvantages of this approach such as the hardware implementation will maximize risks because GSM technology comes into picture. The second method is using J2ME coding for socket connection thus the phone will act as a client and the remote computer becomes the server. A graphic user interface (GUI) is provided for entering the desired communication details like IP address and Port number of the server machine. The router settings of this network should be such that port forwarding enabled. To have a better understanding about these techniques we also compared it with the working of VNC tool. This enabled us to have a comparative study of the implementation methodology used in the various techniques considered. Nowadays, such applications being not just a luxury it is a big step towards achieving bigger tasks like home automation system and Robotics, too.

Keywords: Ubiquitous Computing, wireless communication, client-server communication, J2ME, socket programming, network programming, Wi-Fi.

I. INTRODUCTION

In this paper we have explained the different methodologies for implementation of an application of Ubiquitous Computing for Computer Access through Mobile Phone. Ubiquitous computing (ubicomp) is a post-desktop model of human-computer interaction in which information processing has been thoroughly integrated into everyday objects and activities. In the course of ordinary activities, someone "using" ubiquitous computing engages many computational devices and systems simultaneously, and may not necessarily even be aware that they are doing so. This model is usually considered advancement from the desktop paradigm. More formally Ubiquitous computing is defined as “machines that fit the human environment instead of forcing humans to enter theirs”.

The person/user wanting to control the remote device will be located in any Wi-Fi hotspot region with a J2ME enabled mobile phone. The target device (which is to be controlled) will be a PC at home in any networking environment, either LAN or Wi-Fi or
others. The goal is to switch on/off the PC and access it for further use like downloading files, accessing data or even running an application online or otherwise.

Not long ago, the ability to whip out your trusty PocketPC [2] and set up remote computer access to your desktop PC would have all sounded like science fiction. Today, it’s not only a reality but also practically feasible and highly useful.

In present-day scenario, it is indeed a necessity, and not just a luxury anymore of operation of remote devices using a mobile phone which each of us carries. We have decided to thus, implement the same using computer networking technologies.

II. REVIEW OF LITERATURE

A. Java 2 Platform Micro Edition (J2ME)

Java Platform, Micro Edition, or Java ME, [1] is a Java platform designed for embedded systems (mobile devices

are one kind of such systems). Target devices range from industrial controls to mobile phones (especially feature phones) and set-top boxes. Java ME was formerly known as Java 2 Platform, Micro Edition (J2ME). Java ME was designed by Sun Microsystems, now a subsidiary of Oracle Corporation; the platform replaced 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.

As of 22 December 2006, the Java ME source code is licensed under the GNU General Public License, and is released under the project name phoneME. As of 2008, all Java ME platforms are currently restricted to JRE 1.3 features and use that version of the class file format (internally known as version 47.0).

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 *configurations*, of which there are currently two: the Connected Limited Device Configuration (CLDC) and the Connected Device Configuration (CDC).

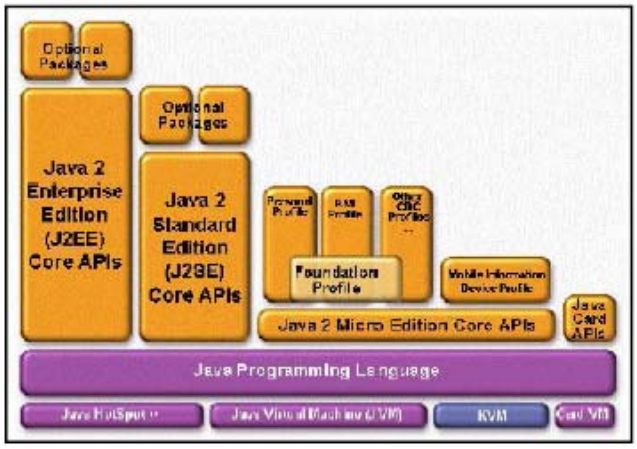

Figure 1. Developing Wireless Applications using the Javaä 2 Platform, Micro Edition.

Several configurations and device profile specifications are now available, including our focus here, the J2ME CLDC and the Mobile Information Device Profile (MIDP).

The above figure [fig.1] gives a diagrammatical view of the components of Java Platform Micro Edition. There are some interesting development and packaging issues raised by the extremely small footprints (memory, CPU, etc.) of mobile and wireless devices. How do you fit useful applications into resource constrained devices? As a developer, you know that you can't write a full-blown desktop application and load it into a phone with a Java Virtual Machine and expect it to work — it just won't fit. Even if it *does* fit, it will be painfully slow, as you are using all sorts of things that aren't fast on these constrained mobile devices.

J2ME addresses this problem by providing for a standard subset of the Java 2 Platform. (When I say "Java 2 Platform" here, I am properly referring to the Java 2 Platform, Standard Edition (J2SE). The J2SE is basically a superset of the J2ME. Java technology for servers, the Java 2 Platform, Enterprise Edition (J2EE) is in turn a superset of the J2SE.) Portability of your application and service code across various J2ME-based devices is insured through the use of J2ME Configurations and Profiles.

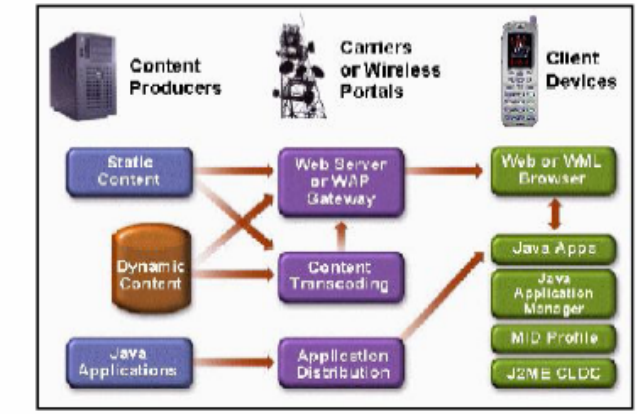

Figure 2. Developing Wireless Applications using the Javaä 2 Platform, Micro Edition

At the top left of the [fig.2], you have content that comes out of simple, static files. Dynamic content is also generated in application servers and Web servers, largely based upon information extracted from a database and being modified dynamically using Java Servlets, Enterprise JavaBeans (EJBs), or other server-side technologies. Increasingly, we see Java applications that sit on a Web server and move through the network to mobile devices. A Web server and/or WAP gateway system will distribute this content out onto the mobile phone or other device.

Again, increasingly we see Java applications moving out across the network, and there is an issue of distribution — how do the Java applications actually move through the network to the phone, how will operators and providers bill for applications and services, etc.? A number of companies are involved with defining this distribution technologies and

business models, and defining the services that should be available.

Anand Ranganathan et al [4] have considered the problem of migrating applications across different ubiquitous computing environments. They introduce the notion of polymorphic applications, where applications can change their structure in order to adapt to different environments. This enables users to perform the same tasks as they move from one environment to the next, seamlessly. They make use of ontologies to ensure that the initial and final structures of a migrating application are semantically similar in terms of functionality and behavior. This paper describes our framework for enabling mobile polymorphic applications. Bill Day [3] discusses the fundamentals of Java technology
for mobile devices and wireless application development. The author gives an overview of the Java 2 Platform Micro Edition

(J2ME), and discusses J2ME Configurations and Profiles used to develop Java applications for mobile phones, PDAs, and twoway pagers. this paper gives you an overview of the technologies that will help you to get started developing wireless applications using J2ME. As mobile phone handsets attain increasing capabilities, Shwetak N. Patel and et al [5] sees many more opportunities for novel applications development. These handsets are typically characterized as constrained computing platforms, due to limitations in computing, storage and interface capabilities. Specialized development environments, such as J2ME, allow for cross-platform development that respects these limitations. While
it is important to respect these resource constraints, the authors also want to highlight some of the unique features of mobile phones, such as high quality audio, constant connectivity and comfortable form factor for use as device to interact with the physical world. In this paper, the authors discuss the J2ME development environment for current mobile phones and demonstrate applications they have developed that respect the resource constraints while simultaneously exploiting the unique features of these commercially available devices. This paper is intended to whet the appetite of potential developers and designers for this important emerging platform.

B. Overview Of Wireless Communication

Wireless communications [1] is a huge field, encompassing everything from radio and television broadcasting through
pagers, mobile phones, and satellite communications. The field of mobile phones is expanding very fast at the same time that standards and protocols are being adopted, used, updated, and sometimes discarded. The other rapidly expanding part of the wireless world is that of wireless local area networks (LANs). Driven by widespread acceptance of the IEEE 802.11 standard, wireless local networking for computers and other devices is spreading rapidly.

Although wireless may seem like a special case, it is actually more intuitive and more natural than wired networking. Someday soon the need to plug a laptop into a network physically will seem quaint and antiquated. The notion that you could walk into a room with your cell phone and have it unable to interact with other devices in the room will seem unbelievably primitive. The future will reveal that *wired* networks are the special case.

Conceptually, wireless communications can be split into two types, *local* and *wide area*. A local device is similar to a key fob with a button that unlocks a car, a 900 MHz cordless phone, a radio control toy, or a Bluetooth network. All of these devices
operate over short distances, typically just a few meters.

Wide area wireless devices operate effectively over a much greater area. A pager or mobile phone is a good example; you can talk on your mobile phone to any other phone on the planet. These devices' greater range relies on a trick, however: a more elaborate land-based network. A mobile phone doesn't have that much more radio power than a radio control toy. What it does have is a network of carefully placed radio antennas (cell towers); the phone can continue to operate as long as it is within range of at least one tower. The mobile phone device receives
service from a wireless *carrier*, a company that operates the land-based network.

While a number of industry consortia and standard bodies, such as the International Telecommunication Union, are trying to define or foster the development of standards for the wireless world, today's wireless world is still fragmented and complex. If you buy a mobile phone in the U.S. today, it might run on Motorola's iDEN network or Sprint's PCS network. Take it overseas to Europe and you'll be out of luck--your phone will not work with Europe's GSM network, nor will it work with the PDC network or any of the other mobile networks that live in Japan.

C. Client Server Communication

In computing, **network programming**[1], essentially identical to socket programming or client–server programming,
involves writing computer programs that communicate with other programs across a computer network. The program or process initiating the communication is called a client process, and the program waiting for the communication to be initiated is the server process. The client and server

processes together form a distributed system. The communication between the client and server process may either be connection-oriented (such as an established TCP virtual circuit or session), or connectionless (based on UDP datagram) A program that can act both as a client
and a server is based on peer-to-peer communication.

Sockets are usually implemented by an API library such as Berkeley sockets, first introduced in 1983. Most implementations are based on Berkeley sockets, for example Winsock, introduced in 1991. Other socket API implementations exist, such as the STREAMS-based Transport Layer Interface (TLI).

These are examples of functions or methods typically provided by the API library:

- Socket ( ) creates a new socket of a certain socket type, identified by an integer number, and allocates system resources to it.
- Bind ( ) is typically used on the server side, and associates a socket with a socket address structure, i.e. a
  specified local port number and IP address.
- Listen ( ) is used on the server side, and causes a bound TCP socket to enter listening state.
- Connect ( ) is used on the client side, and assigns a free local port number to a socket. In case of a TCP socket, it
- causes an attempt to establish a new TCP connection.
- Accept ( ) is used on the server side. It accepts a received incoming attempt to create a new TCP connection from the remote client, and creates a new socket associated with the socket address pair of this connection.
- Send ( ) and recv ( ), or write ( ) and read ( ), or recvfrom ( ) and sendto (), are used for sending and receiving data to/from a remote socket.
- Close ( ) causes the system to release resources allocated to a

Eclipse [1] is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java and can be used to develop applications in Java and, by means of various plug-ins.

III. METHODOLOGIES

A. Hardware Approach

Using hardware components like microcontroller to establish connection between any phone and the computer. There is a cell-phone required with Network Service provider to establish the connection with the remote device which is to be automated. The device to be automated will be connected to a Modem (having a SIM card). This will receive the appropriate requests.

Intelligence will be provided to this modem using a PCB mounted with a Microcontroller. The necessary connections will be made on PCB. Embedded C codes will be executing on the device that will be transferred on the chip for its functioning. Simultaneously, VB codes will start running on the computer once it receives the power-on request.
once it receives the power-on request.

Software platform details:

- Embedded C
- Visual Basic

Hardware platform details:

- Microcontroller 8051/8091.
- Modem with service providing SIM card.
- Cell-phone.

Drawbacks of this approach are:

- Additional hardware requirement makes the system unfeasible for practical implementation in the professional environment.
- Not only will companies oppose to install so much hardware, but also make working difficult, because the user will have to be well versed with Microcontrollers and embedded coding, AT Commands etc.
- Security is the biggest concern these days and hardware implementation will maximize risks because GSM technology comes into picture.

B. Revolutionizing The Approach

We will use the networking techniques instead of hardware approach to overcome the above drawbacks. The magic packet is sent on the data link or layer 2 in the OSI model and broadcast to all NICs using the network broadcast address; the IP-address (layer 3 in the OSI model) is not used.

The *magic packet* is a broadcast frame containing anywhere within its payload 6 bytes of all 255 (FF FF FF FF FF FF in hexadecimal), followed by sixteen repetitions of the target computer's 48-bit MAC address, for a total of 102 bytes. It is typically sent as a UDP datagram.

Drawbacks of this approach are:

- Does not provide a delivery confirmation.
- May not work outside of the local network.
- Broadcast packets are generally not routed.

C. Implementation

The person/user wanting to control the remote device will be located in any Wi-Fi hotspot region with a J2ME enabled mobile phone. The target device (which is to be controlled) will be a PC at home in any networking environments, either LAN or Wi-Fi or others. The goal is to switch on/off the PC and access it for further use like downloading files, accessing data or even running an application online or otherwise.

The mobile phone will act like a client in the communication and the remote PC will act like the server. As the phone is J2ME enabled we shall request for establishment of connections using socket programming. The corresponding port numbers and address will be specified in the programs. The request will be sent over Wi-Fi (as the mobile phone user will be in Wi-Fi hotspot). The router settings of this network should be such that port forwarding enabled.

The remote PC that we intend to control can be connected over any internet connection (i.e. LAN, Wi-Fi, etc.). This PC will have an API being executed for incoming request. This PC will act as a server with a live IP address. The port number of the client will be specified to initiate connection establishment. Once the connection is established the J2ME implementation on the phone will be used to access the PC in the remote location.

Drawbacks of this approach are:

- Port Forwarding: The request is sent over Wi-Fi (as the mobile phone user will be in a Wi-Fi hotspot). The router settings of this network should be such that port forwarding is enabled.
- Identifying the Server over the network: Hard coded into the client program requires that the user identifies the server. Also read from a configuration file and use a separate protocol/ network service to look up the identity of the server.
- Services and ports: Many services are available via "well known" addresses (names). There is a mapping of

service names to port numbers.

IV. BLOCK DIAGRAM

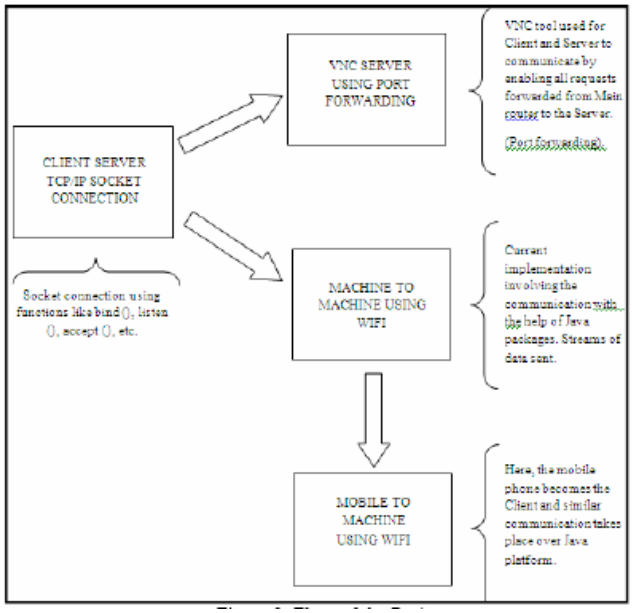

Figure 3. Flow of the Project

V. IMPROVED METHODOLOGY

The software used is Eclipse Version 3.3 which is a Multilanguage software development environment comprising an
integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java and can be used to develop applications in Java and, by means of various plug-ins. Also, the coding is mainly being done in Java language with the creation of namely two packages i.e. Remote Server and Remote Client, communicating over Wifi network. There is soft binding of port nos. at present, which we shall change in the future.
The Remote Client package consists of following executable applications-

- Client Initiator: For initiating request to Server on a given network. This is the first and most important step for beginning any network connection.
- Screen Spyer: For spying the screen of the Server postconnection. This will enable us full control of the remote device.
- Enum Commands: This includes mouse over and sensor commands which have enumerated data types.
- Server Delegate: Once the connection is established and control is possible, this enables in starting other complex requests like a file transfer or running an application on remote device etc. On the other hand, The Remote Server package consists of following executable applications.

- Server Initiator: For initiating a broadcast request to Clients on a given network. This is the first and most important step for beginning any network connection.
- Client Screen Spyer: For spying the screen of the Client post-connection. This will enable us full capture the events at the Clients side for applications as well as security purposes.
- Client Handler: This helps in basically handling the Client's requests as well as running applications.
- Enum Commands: This includes mouse over and sensor commands which have enumerated datatypes.
- Client Commands Server: Once the connection is established and control is possible, this enables in starting other complex requests like a file transfer or running an application on remote device etc.

VI. CONCLUSION

Considering the various approaches available to achieve our aim of remotely accessing a PC through mobile phone,
We decided to implement a networking approach. The main advantages of this project are

There is no external hardware requirement.

It is platform independent i.e. it can be implemented on different connection technologies and topologies like
LAN, Wi-Fi, etc.

The user can be a layman without any technical knowledge.

The above methodology has been successfully implemented using two computers with user defines port numbers. However hard coding the port numbers will be more efficient in terms of performance as well as time. We need to integrate the same on a Java enables phone.

In terms of scalability, when the number of users of our application increases the use of dynamic IP addresses will become a necessity. This is because, when the remote PC is not being accessed by any user it will lead to wastage of available IP address.

The phone will act as a client and the server will be a remote computer. The client package that has been executed on a client machine (computer) will be transferred to a mobile phone where it will execute as a mobile application. A graphic user interface will be provided for entering the desired communication details like IP address and Port number of the server machine.

The server package will be simultaneously executing on the server machine and it will be ready to accept and correspond to incoming requests from client phone.

This concept can be extended to various other intelligent devices at home thus transforming the concept into a home Automation system.

VII. REFERENCES

[1.] Wikipedia- http://en.wikipedia.org/wiki/Wake-on-LAN
[2.]http://www.makeuseof.com/tag/set-up-remote-computeraccess-with-your-mobile
[3.] Bill Day, "Developing Wireless Applications using the Javaä 2 Platform, Micro Edition", Technology Evangelist Sun
Microsystems, Inc., 2001.
[4.] Anand Ranganathan, Shiva Chetan and Roy Campbell, "Mobile Polymorphic Applications in Ubiquitous Computing Environments", Department of Computer Science- University of Illinois at Urbana-Champaign, 2004.
[5.] Shwetak N. Patel and Gregory D. Abowd, "Beyond Mobile Telephony: Exploring Opportunities for Applications on the Mobile Phone Handset", College of Computing & GVU Center, 801 Atlantic Drive, Atlanta, GA 30332-0280, USA.