Project Report

On

Bluetooth Scatternet

Ananth V
Ravi D Suvarna
Sandeep K Jain

© Open Source

(https://scatternet.dev.java.net)
# Table of Contents

1. **Introduction** ....................................................................................................................... 6  
   1.1 Purpose ............................................................................................................................... 6  
   1.2 Scope ................................................................................................................................. 6  
   1.3 What this application is? ................................................................................................. 6  
   1.4 Motivation .......................................................................................................................... 7  
   1.5 About Bluetooth ............................................................................................................... 8  
   1.6 Literature Survey .............................................................................................................. 11  

2. **Software Requirement Specification** ................................................................................. 14  
   2.1 Overall Description .......................................................................................................... 15  
      2.1.1 Product Perspective: .................................................................................................. 15  
      2.1.2 Product Functions and Features ............................................................................... 17  
   2.2 Specific Requirements ..................................................................................................... 18  
      2.2.1 Functional requirements ......................................................................................... 18  

3. **High Level Design** ............................................................................................................ 21  
   3.1 Architectural strategies .................................................................................................... 22  
   3.2 External Interfaces .......................................................................................................... 22  
      3.2.1 User Interfaces .......................................................................................................... 22  
      3.2.2 Hardware Interfaces ............................................................................................... 28  
      3.2.3 Software Interfaces ................................................................................................. 28  
      3.2.4 Communication Interfaces ..................................................................................... 28  

4. **Detailed Design** .................................................................................................................. 29  
   4.1 Design Considerations ...................................................................................................... 30  
      4.1.1 Assumptions and Dependencies ............................................................................. 30  
      4.1.2 General Constraints ................................................................................................. 30  
      4.1.3 Architectural strategies ............................................................................................ 31
4.2 System Architecture ........................................................................................................... 32
  4.2.1 Use Case diagram: ........................................................................................................ 33
  4.2.2 Class diagram: ............................................................................................................... 35
  4.2.3 Flow Chart ..................................................................................................................... 37

4.3 Network Architecture ......................................................................................................... 38
  4.3.1 Piconet .......................................................................................................................... 38
  4.3.2 Scatternets ..................................................................................................................... 40
  4.3.3 RIP - Routing Information Protocol ............................................................................... 41

5 Implementation ....................................................................................................................... 43
  5.1 Tools and technologies used ............................................................................................. 44
    5.1.1 Netbeans ....................................................................................................................... 44
    5.1.2 Sun Wireless Toolkit (WTK): ...................................................................................... 45
    5.1.3 Bluetooth stacks .......................................................................................................... 46
    5.1.4 MYSQL: ........................................................................................................................ 47
    5.1.5 BLUECOVE: ............................................................................................................... 48
    5.1.6 J2SE : (Java 2 Standard Edition) ................................................................................. 49
  5.2 Source code listing .............................................................................................................. 51

6 Testing ..................................................................................................................................... 53
  6.1 Testing Scatternet Application ............................................................................................ 54

Conclusion .................................................................................................................................... 56

Future Enhancements ................................................................................................................. 56

Applicability ................................................................................................................................ 56

References ..................................................................................................................................... 58
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Bluetooth Stack Architecture</td>
<td>10</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Product Perspective</td>
<td>16</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Application Frontend</td>
<td>23</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Application Update Database</td>
<td>24</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Application Send Message</td>
<td>25</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Application Received Message</td>
<td>25</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>System Architecture</td>
<td>32</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Class Diagram</td>
<td>35</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Flow Chart</td>
<td>37</td>
</tr>
</tbody>
</table>
Chapter 1
Introduction
1. **Introduction**

Bluetooth is a promising new short range wireless technology, which enables portable devices to form PAN (personal area networks) or Scatternet and is based on a frequency hopping physical layer. This implies that hosts communicate after having discovered each other by synchronizing their frequency hopping patterns. This also means that even if all nodes are within direct communication range of each other, only those nodes which are synchronized with the transmitter can hear the transmission. To support any-to-any communication, nodes must be synchronized so that the pairs of nodes (which can communicate with each other) together form a connected graph.

Bluetooth

A *Scatternet* is an ad hoc network created by interconnecting several Bluetooth *piconets*. Each piconet uses a different radio channel constituted by a frequency hopping code. The way the devices are grouped in different pioneers and the way the piconets are interconnected greatly affect the performance of the Scatternet in terms of capacity, data transfer delay, and energy consumption.

1.1 **Purpose**

The purpose of this application lies in establishing a Scatternet between in-range and out of range Bluetooth enabled devices, thus enabling different application systems to exchange data over an extended range.

1.2 **Scope**

The Bluetooth Scatternet has its advantages as any other ad-hoc network. Scatternets have the potential to bring the interconnectivity of the Internet to the physical world through wireless devices.

1.3 **What this application is?**

This Scatternet application has been developed primarily, not as a texting or a product information service, but as an elegant demonstration of how a Scatternet...
can be formed at the application level.

1.4 Motivation

The ubiquitous use of information intensive consumer devices such as cell phones, personal digital assistants (PDAs), and laptop computers have called for a new networking paradigm for interconnecting them. The goal is to create a personal area network (PAN) that accommodates seamless information transfer between different devices with varying capacity in an *ad hoc* manner without the need for manual configuration, cables, or wired infrastructure.

1.5 About Bluetooth Wireless Technology:

Bluetooth wireless technology is an open specification for a low-cost, low-power, short-range radio technology for ad hoc wireless communication of voice and data anywhere in the world.

**Attributes:**

- An open specification means that the specification is publicly available and royalty free.
- Short-range radio technology means devices can communicate over the air using radio waves at a distance of 10 meters (m). With higher transmission power the range increases to approximately 100 m.
- Communication is within a short range, and hence the radios are low power and are suited for portable, battery-operated devices.
- Bluetooth wireless technology supports both voice and data, allowing devices to communicate either type of content.
- Bluetooth wireless technology works anywhere in the world because it operates at 2.4GHz in the globally available, license-free, industrial, scientific, and medical (ISM) band.
History of Bluetooth wireless technology:

Bluetooth communications originated in 1994, when Ericsson began a study to find alternatives to connecting mobile phones to its accessories. The engineers looked at a low-power and low-cost radio interface to eliminate cables between the devices. However, the engineers also realized that for the technology to be successful, it had to be an open standard, not a proprietary one.

In early 1998, Ericsson joined Intel, IBM, Nokia, and Toshiba and formed the Bluetooth Special Interest Group (SIG) to focus on developing an open specification for Bluetooth wireless technology. The original companies, known as promoter companies, publicly announced the global Bluetooth SIG in May 1998 and invited other companies to join as Bluetooth adopters in return for a commitment to support the Bluetooth specification. In July 1999, the Bluetooth SIG published version 1.0 of the Bluetooth specification. In December 1999, four new promoter companies—3Com, Agere, Microsoft, and Motorola—joined the Bluetooth SIG.

Since then, the awareness of Bluetooth wireless technology has increased, and many other companies have joined the Bluetooth SIG as adopters, which give them a royalty-free license to produce Bluetooth-Enabled products.

Why this technology is called Bluetooth wireless technology?

It was named after a Danish Viking king, Harald Blatand, who ruled circa A.D. 940-981. Blatand loosely translates to “Bluetooth.” During his reign, king Harald Blatand is supposed to have united and controlled Denmark and Norway. Because this new radio technology was expected to unify the telecommunication and computing industries, it seemed fitting to name it after King Harald. A part-time historian on the team proposed Bluetooth as the internal code name. Because the Bluetooth SIG marketing team could not come up with a better name that was not already trademarked, the name stuck.
Bluetooth Vision:

Bluetooth wireless technology was originally developed as a cable replacement technology for connecting devices such as mobile phone handsets, headsets, and portable computers with each other. However, wireless connectivity between fixed and mobile devices enables many usage scenarios other than cable replacement. By enabling wireless links and communication between devices, a short-range wireless network was created that gave rise to the notion of a personal area network. Designed as an inexpensive wireless networking system for all classes of portable devices.

There are many usage scenarios for Bluetooth wireless technology, and the technology has been put to wide use. Couple of usage models.

- The three-in-one phone usage model allows a mobile telephone to be used as a cellular phone in the normal manner, as a cordless phone that connects to a voice access point and as an intercom or “Walkie-Talkie for direct communication with another device. The cordless telephony and the intercom features use Bluetooth wireless technology.
- The second use case is wireless telematics. Assume that a user who is talking on a cell phone conversation in the hands-free mode. Using Bluetooth communication the user can continue the phone conversation using the microphone and speakers equipped in the dashboard of the automobile.
- Another use case is the instant postcard, whereby a user with a digital transmits a photo via a data access point, which could be a mobile phone.
Overview of Bluetooth Stack Architecture

The Bluetooth protocol stack can be broadly divided into two components: the Bluetooth host and the Bluetooth controller.

The Host Controller Interface (HCI) provides a standardized interface between the Bluetooth host and the Bluetooth controller.

The Bluetooth host is also known as the upper-layer stack and usually is implemented in software. It is generally integrated with the system software or host operating system. Bluetooth profiles are built on top of the protocols. They are generally in software and run on the host device hardware.
The Bluetooth radio module or controller usually is a hardware module like a PC card that plugs into a target device. More and more devices have the Bluetooth radio module via the HCI. The Bluetooth radio module usually interfaces with the host system via one of the standard input/output mechanisms, such as the universal asynchronous receiver-transmitter and universal serial bus.

1.4 Literature Survey

Given below are a few ideas that have been borrowed from research papers.

**Yong Liu@etal[2003]:** Most Scatternet formation algorithms ignored the “standby” state of Bluetooth which tended to interconnect all devices as a big Scatternet and maintained its full connectivity. Yong Liu and co suggest an on demand formation to utilize power efficiently. This can be done by either by 1.) Flooding based route discovery or 2.) backward formation of Scatternet route. There are three possible broadcast schemes. They are L2CAP, LMP and EID. We shall not go into detail concerning these mechanisms but the most efficient of these is the EID. In this, the master changes its frequency train every 2.56 secs and each neighbor scans the enquiry channel in a maximum period of 2.56 secs. The inquiry message can arrive at all neighbors within 5.12 secs. Now we move on to the next mechanism. In the backward formation of Scatternet route, the destination responds immediately with a RRP, which starts the Scatternet formation process along the newly discovered route. Each device in the new route simply pages its upstream neighbor, creates a physical channel in between and delivers the packet.

**Chiara Petrioli@etal[2003]:** Chiara Petrioli and co propose the formation of the scatternet using what are known as bluestars. The protocol involves three phases, topology discovery, bluestars formation and the configuration of the bluestars into a “blue-constellation” i.e. the connected scatternet.

1) The first phase concerns the discovery of neighboring devices. Our protocol relies on the mutual, “symmetric” knowledge of neighboring devices, which means that if node
v knows node u, u must also know v. By alternating between inquiry and inquiry scan modes and by establishing temporary piconets, each device discovers neighboring devices and, at the same time, it makes them aware of its presence.

2) The second phase takes care of piconet formation: One master and some slaves set up communication links to form a bluestar. Based on a locally and dynamically computed weight, each node decides whether it is going to be a master or a slave.

3) The final phase concerns the selection of gateway devices to connect multiple piconets so that the resulting blue-constellation is connected. By using the information gathered during the bluestars formation phase, each master selects some of its slaves to be gateways to neighboring piconets.

_Patricia McDermott Wells[2004]:_ The author in this paper shares her ideas on some scatternet models reviewed here for their suitability for dynamic adhoc networks. They are bluetrees, TSF and bluestars.

1) Bluestars: Here the discovery and the subsequent path establishment is done in the form of a “blue” rooted tree. The path is typical of a n-ary tree. There exists just one path between each peer and any removal of a node has to be dealt with immediately to ensure no loss of connectivity.

2) TSF: Tree scatternet formation consists of one or more rooted spanning trees, which autonomously and continuously attempt to merge into a topology with fewer trees. This is a decentralized process, in that each tree continuously searches for other trees and the nodes communicate with and maintain information about adjacent nodes only.

3) We shall not go into the details of bluestars as all necessary information have been discussed.

The project is based on some key ideas that have been discussed so far and implement them in trying to setup a dynamic and robust Scatternet adhoc model. The main aim is to combine the backward formation of a Scatternet route mechanism and the concept of the blue-constellation. The steps are demonstrated below:
1) The node which needs to transmit the data takes the initiative to be the master and sends a RDP to each of its immediate neighbors. The neighbors then discover each of its immediate neighbors and send the RDP further by attaching their address to it.

2) The destination then sends this packet in the reverse direction in which it came so that it reaches the master. Similarly all the nodes do the same thing and in due time the master knows all the paths to every node in the vicinity (Route Discovery).

3) Now to transmit any data, all that the master has to do is to attach the route to the data and send it over the network. Only those nodes in the path will transmit the packet further. There will be no acknowledgement sent back after the packet reaches the destination.

4) After the master sends the data over the network, the network can be “dismantled” or if any other node needs to be the master, the previous master shares its route data with it.
Chapter 2

2 Software Requirement Specification
2.1 Overall Description

To establish a Scatternet, we require the constant communication between devices in order to obtain the information (Address, Friendly name) of devices in-range and devices out of their range. Once this information is available, the route can be established. Here, certain devices take the role of a server and some as clients. However, role reversal is certainly possible and required. To aid this role reversal we have a server thread running in the background always but a client thread is created only when required. For routing to occur, intermediate mobiles do act as routers.

Let us now have a brief look at our application. The Scatternet application consists of two components.

- **PC Application:** Runs the Scatternet application on a PC. It has a database integration used also as a product information server.
- **Mobile Application:** Runs the Scatternet application on a mobile device. Acts as a text client and server.

### 2.1.1 Product Perspective:

This Application enhance small business where advertising or attracting a customer play s major role. With help of Scatternet a user may attract customer free of cost using Bluetooth. Let us consider a business establishment of various products as shown in the figure below. Every stall has a Scatternet enabled Bluetooth connection.
A customer walks in and has a difficult time assessing the stores present and the one which would provide him with the best favorable price for his item, a pendrive. So all he has to do is to key in the words “search pendrive” and send it to the nearest available Bluetooth device. The search is done throughout the Scatternet and the product matching his specification is returned with the price attached and the store which provides it. Thus saving him a lot of effort, time and may be money. Then the customer selects the best suitable store and walks in to purchase the specific pendrive which suits his requirement. The above use case scenario is one of the best example for how the Scatternet takes place, thereby providing a convenient way of sales at commercial places.
2.1.2 Product Functions and Features

This product is an implementation of Scatternet at application layer. It features Scatternet establishment between interconnected devices. It features several functions like searching devices, searching services, path establishment, generating routing table information, enables user to search for any product information or provide text messaging service in order to help with each other in the network.

User Characteristics

Scatternet shall be designed as a tool for the casual user. A casual user shall be defined as one possessing general knowledge of the Microsoft Windows operating system and general knowledge of using the Bluetooth enabled Cell phone. The casual user must have understanding of Scatternet or Bluetooth ad-hoc network to use this application.

General Constraints

The Scatternet application shall be designed to run on any PC or 100% compatible computer with a Pentium-class or comparable processor. The host computer shall have at least 256 megabytes of RAM installed, although 1024 megabytes of installed RAM is preferable. There shall be 20 megabytes of free disk space. The host computer shall run Microsoft Windows XP or newer. The GUI shall be implemented primarily in Java.

Assumptions and Dependencies

The user application shall have full access rights connectivity resources. The customer or the customer's network administrator shall have permission to install Scatternet database on the user's workstation.
2.2 Specific Requirements

2.2.1 Functional requirements

There are some main functional requirements for establishing a Scatternet.

- Scatternet runs as a Platform Independent application.
- Scatternet shall handle multiple client requests.
- Scatternet shall maintain any web link database that it creates i.e it shall be have the ability to add, delete, or modify database records.
- Scatternet user interface shall permit the user to send message to another device in Scatternet network.
- Scatternet user interface shall permit the user to enter keywords to retrieve data from the Bluetooth link database.
- Scatternet user interface shall permit the user to view received data.
- Scatternet shall update its routing information when necessary and required.
- Scatternet enabled device shall act as a router for other devices.

Design Constraints

Design constraints include restrictions and limitations that constrain the implementation of this software application. It includes user interface, display and performance, safety, reliability, scalability, and portability.

Software Constraints

The following are the software constraints of the Bluetooth Scatternet application:
• The Bluetooth Scatternet shall be implemented in java using Netbeans IDE and the application will be saved as java archive (*.jar) format (executable).

• The Bluetooth Scatternet search engine and database interface shall be implemented using java, MySql, MySql library.

• The Bluetooth Scatternet midlet shall be implemented in java using Net beans IDE should run on the Sun wireless toolkit.

• Bluecove, BlueZ stacks, BlueZ-libs-devel shall be used for the application development.

  ➢ Bluecove: Bluecove is a JSR-82 implementation for various Bluetooth stacks.
  ➢ BlueZ stacks: BlueZ stack is a Bluetooth stack implemented in most of the Linux distributions.
  ➢ BlueZ-libs-devel: BlueZ-libs-devel contains header files for use in Bluetooth applications. It also contains development libraries for Bluetooth applications.

**Hardware Constraints**

The following are the hardware constraints of the Bluetooth Scatternet application:

• Bluetooth Scatternet requires 2 or more piconets to be established with active Bluetooth connection. Piconet is a network of Bluetooth devices in accessible range.

• Bluetooth Scatternet requires devices of Series 40 in specific.

• We require a Bluetooth adapter a.k.a dongle for enabling Bluetooth connectivity in PCs and notebooks if required.
Bluetooth dongle (Adapter): Bluetooth dongle enable short range wireless data- connectivity between Bluetooth enabled devices. Bluetooth dongle technology enables your PC or notebook for point to multi-point friendly connectivity with other Bluetooth devices, such as PDA, mobile phone.

**Network Constraints**

- The target hardware system shall have the ability to establish and maintain a reliable Bluetooth connection.
- The connection establishment between devices in a Scatternet is slow due to the large number of hops sometime involved and the small bandwidth available in Bluetooth 2.0.
- At any instance of time, the device can send data to only one device.
Chapter 3

High Level Design
3.1 Architectural strategies

Here we take a look at the decisions and strategies taken for the development of the Scatternet application.

Platform preferred was the java 2 micro edition for its extensive library and java itself for its multithreaded programming support. MySQL was used for the creation and maintenance of the product information database.

Bluecove, the JSR-82 library was also used. This supports the blueZ Bluetooth stack found in almost all Linux distributions.

We have not used any extensive memory management policies as such. Java’s automatic garbage collecting system will be sufficient.

The product information database is hosted on a web server to give the advantage of universal access. The PC application contacts the web server to retrieve the product information.

3.2 External Interfaces

This section describes Bluetooth Scatternet external interface requirements.

3.2.1 User Interfaces

This section describes Bluetooth Scatternet user interface requirements.
3.2.1.1 Introduction

The user interfaces of this PC system are the computer screen, the keyboard, and the mouse. The computer screen is used to display the bts output screens. The keyboard and mouse are used to retrieve user inputs.

The user interfaces of this mid system are the Led display, the keypad, and the joystick the led screen is used to display the bts output screens. The keyboard and mouse are used to retrieve user inputs.

3.2.1.2 User Interface:

User Interface map for both pc and mobile application are described below:

PC application main screen is divided into following three tab pane s

1. The first tab is the Scatternet server tab. It has a start, stop, an update Product Information button and an exit button. This Scatternet application contacts a web server and retrieves information for a product and displays it by sending the information over Bluetooth to devices available through the Scatternet.

![Figure 3.1 Application](image)
Figure 3.2 Application Update Database

Figure 1.2 also shows the window that is opened when the update button is clicked. This may take several seconds as it has to contact the web server and then retrieve the data.

The window opened gives you the option of adding, deleting and editing the entries in the database. You can then save your changes. Although you might note that the role of My Sql is minimal here. It is however just a demo of how it can be used to store and then retrieve data. This may also be video and audio files which are much larger. We are using a web server here because of the flexibility associated with it. This application retrieves the same data wherever it is run.

2. The second tab is the send message tab. This is in case you have to send any other message other than product information
3. Figure 1.3 shows the send message tab. It has a pop down menu for selecting the device to which the message has to be sent. It also has a text box where you can enter the message that is to be sent.

4. The third tab is the receive message tab which displays the message that has been received.

Figure 1.4 shows the received message tab. Any message that it receives will be displayed here.

*Note:* If the message received is prefixed with ‘Search’, the product database will be searched keeping the message as an index and information of the product title matching it will be sent to the device, thus acting as a product information application.
The main screen of mobile application and its operations are described below:

![Figure 3a](image1.png)  ![Figure 3b](image2.png)

Figure 2.1b shows the initial screen of the application when launched. It has four options. A “send new message”, “received message”, “rescan devices” and an “exit” option.

- **Send New Message**: This option when chosen opens up a text box in which the message can be written (Figure 2.2a). This can be sent with the send option present on the bottom right of the screen. Then the devices list is displayed to which the message can be sent. This device list contains both the in-range devices and the devices not in range. If the device selected is in-range, the message is delivered directly and if not the message is sent through a pre-determined route of the Scatternet. However it should be noted that the devices should be given sufficient time to exchange routing information and determine the ‘reachable’ devices.
- **Received Message**: The received message option works as an inbox of messages. On choosing this option, we get the inbox which displays the various devices from which messages have been received (Figure 2.2b). The corresponding text message can be viewed by choosing the device (Figure 2.2c).

- **Re-Scan devices**: This option is to refresh the device list so that any new device that may have entered the range or devices which may have stepped out of range can be detected and duly updated.

- **Exit**: Terminates the application.
3.2.2 Hardware Interfaces

The hardware interfaces used is a Bluetooth dongle if required. Its only used when a Bluetooth adapter is not present, usually in PCs. We also require a java enabled, Bluetooth enabled mobile device where our application can install and run.

3.2.3 Software Interfaces

The software interfaces used are mainly a Bluetooth stack and the Bluetooth API’s for development. The Bluetooth stack preferred for the Scatternet application is the BlueZ stack, but widcomm stack would also pose no problems.

3.2.4 Communication Interfaces

The communication interface used is the Bluetooth frequency hopping spectrum.
Chapter 4

4    Detailed Design
4.1 Design Considerations

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution.

4.1.1 Assumptions and Dependencies

It is assumed that end user using this software may either run this application on pc or his cell phone. User running on pc must have Bluetooth dongle with Microsoft or widcomm or bluez stack and data base to store and retrieve data.

And user running on cell phone must have series 40 operating system.

End mobile user may either want to send or receive messages from other phone or cell phone user retrieve any product information from pc or pc might want to host services.

4.1.2 General Constraints

Describe any global limitations or constraints that have a significant impact on the design of the system's software (and describe the associated impact). Such constraints may be imposed by any of the following (the list is not exhaustive):

- **Availability or volatility of resources**

  It is necessary that Bluetooth stack is not busy providing any other service like dial up obex etc. Bluetooth is switched on and set Discoverable.

- **Security requirements**

  It is necessary that end user had paired neighboring Bluetooth devices so as to reduce time delay. User must also set permission to use connectivity devices
• **Memory and other capacity limitations**

    More the number of devices in Scatternet more the number of entries to be stored in routing table, memory user is directly proportional to number of devices in Scatternet.

• **Performance requirements**

    It is necessary that end user had paired neighboring Bluetooth devices so as to reduce time delay.

### 4.1.3 Architectural strategies

Here we take a look at the decisions and strategies taken for the development of the Scatternet application.

• Platform preferred was the java 2 micro edition for its extensive library and java itself for its multithreaded programming support. MySQL was used for the creation and maintenance of the product information database.

• Bluecove, the JSR-82 library was also used. This supports the bluez Bluetooth stack found in almost all Linux distributions.

The product information database is hosted on a web server to give the advantage of universal access. The PC application contacts the web server to retrieve the product information.
4.2 System Architecture

To have a good understanding of the system architecture, we shall have a look at use case diagrams, flow charts and data flow diagrams. This will give us a thorough outlook of the system architecture.

![Figure 4.1 System Architecture](image)

The above figure depicts a simplistic view and an instance of a Bluetooth Scatternet system. It shows two piconets forming a Scatternet. Every piconet has a server and
clients. The server advertises its services and the clients can access these services. Assuming that there’s one or more devices in common (in range) between these two piconets, this or these devices will act as a bridge between them. Any data exchange between devices in these piconets must pass through the bridge devices. With respect to our application, these servers and clients can be a mobile or a PC.

4.2.1 Use Case diagram:

A use case diagram as the name suggests is an insight into how end users might use the application. The use case diagram for our Bluetooth Scatternet application is shown and explained in the next page.

Let us have a good look at the use case shown above and examine the use cases and the end users. The use cases are:
• Search services: Search for services that exist in your range as well as the Scatternet if one is established. Update the device list with the servers which have advertised their services.

• Send message: Send a message to the selected recipient. If the recipient is more than a hop away, specify the intermediate device to which it has to send first.

• View received message: All received messages are stored in the inbox of the recipient. Like any inbox, it has the sender’s name as the indicator. Selecting it will display the message sent by it.

• Search product database: The user can get information on his desired product by searching the product database. The product database can be searched by prefixing the keyword with search and sending it to the appropriate server. The server searches with the keyword and returns information of the product matching the keyword.

• Send information: This use case is with respect to the product info server. The server after receiving the keyword searches the database with it and returns the result.
4.2.2 Class diagram:

A class diagram shows inter-relationship between the classes or modules of the application. It helps us get a better understanding of the system as a whole.

---

Figure 4.2 Class Diagram

The above diagram is a generalized class diagram of our Bluetooth scatternet application. It has seven classes,

- Scatternet class: Creates the midlet required to run the application. The midlet handles the GUI.
- DisplayHandler class: Handles the display of the entire application. Of both the client and server and the routing table class.
- RoutingTable class: Is responsible for generating the routing table of the device in which the application is running. It is also responsible for exchanging the routing information between the devices.
- BluetoothState class: Checks the current status of the device i.e either the server or the client that is running. This is then set accordingly so that the device can behave according to it.
- BtServer class: Runs the server thread of the application.
- BtClient class: Runs the client thread of the application.
- ClientHandler class: Runs in the server and as the name suggests is responsible for handling the client and its requests.

The Scatternet class is like the main class. It is aided by the DisplayHandler class which handles all the display of the scatternet application including the graphical user interface i.e the midlet. As shown in the class diagram, the DisplayHandler handles the GUI of the Bluetooth client, the Bluetooth server and the routing table. A GUI for the client and server is pretty obvious but the reason why we need the assistance of the DisplayHandler for the routing table is that the routing table is in charge of updating the device list for the application.

The BtServer handles the server connections. It is associated with the Bluetooth state which defines the state it is in, like ready, stopped or running etc. The BtClient handles the client side of the application. It does the service request to the server and gets the job done. It also is associated with the Bluetooth state class.
The figure in the previous page illustrates the flow chart of the Scatternet application in general. It has been simplified to a great extent to get a good understanding of how the system works.

The functionality of the above flow chart can be summed up in the algorithm described below.

Steps:

1. Start
2. Start the Bluetooth client or server as required.
3. Search for devices and services.
4. if search completed goto step 5 else repeat step 3.
5. if any new devices are found goto step 6 or goto 7.
6. Exchange routing information between the new devices.

7. If the user wants to send a message, then select the device and goto step 9 else goto step 8.

8. if the user wants to exit goto step 10 else goto step 7.


10. Exit.

4.3 Network Architecture

4.3.1 Piconet

A connection between one master and up to seven slaves is called a piconet. Such a connection is created when two units are within transmitting and receiving range from each other (ad-hoc connection). In every piconet there can be only one master (the Bluetooth unit that initiates the connection) and it is defined at the time of establishing the first connection to form a piconet. There is no difference in hardware or in the capabilities of the different BT units in a piconet that is defined as slave or master, but there is a need to assign a single master to each piconet for synchronization and control needs. A masters job is to control the frequency hopping mechanism and to set the clock’s frequency by which all salve units in the piconet will be synchronized to. It is also the masters job to manage the order of transmissions inside a piconet. The master notifies each slave of its turn to transmit a packet by calling to this slave in the time slot previous to the slot in which the slave is supposed to transmit, and in this manner sets the transmission order inside the piconet.
From the above figure, it can be understood that a master can function as a master in one piconet only simultaneously. It can, however, function as a slave in other piconets, but at the cost of suspending activity at the network in which he is a master. Slaves have no limitation on participating in a few networks simultaneously and do not affect the functioning of any network.
4.3.2 Scatternets

When a number of piconets cover the same area and reception space and there are some units that are participating in inter-piconet connections a Scatternet is formed.

Two piconet networks connected to form a Scatternet network.

A unit participates in two or more piconets by applying time multiplexing. To participate on the proper channel, it should use the associated master device address and proper clock offset to obtain correct phase. A Bluetooth unit can act as slave in several piconets but as a master only in one piconet (since if two piconets were using a common master they’d actually be synchronized on the same channel and hence are in fact a single piconet).

A master can become a slave in another piconet by being paged by the master of this other piconet. On the other hand, a unit participating in one piconet can page the master or slave of another piconet. Since the paging unit starts out as a master, a master-slave role-switch is required.

As mentioned, time multiplexing must be used to switch between piconets. A unit can request to enter HOLD or PARK mode in piconet A, and during this time it may join
piconet B by simply changing the channel parameters. (SNIFF can also be used if the
sniff slots allow enough time to pass from one piconet to the other).
Since the clocks of two masters of different piconets are not synchronized a unit
participating in two piconets has to keep synchronized with both masters and keep a
record of the channel parameters of both (or more) piconets it participates on.

4.3.3 RIP- Routing Information Protocol

Routing is the task of finding a path from a sender to a desired destination.
The goal of a routing protocol is very simple: It is to supply the information that is
needed to do routing.

Background –Distance Vectors

The Routing Information Protocol, or RIP, as it is more commonly called, is one of
the most enduring of all routing protocols. RIP and the myriad RIP-like protocols
were based on the same set of algorithms that use distance vectors (DV) to
mathematically compare routes to identify the best path to any given destination
address. These algorithms emerged from academic research that dates back to 1957,
and are based on the Bellman-Ford algorithm.
A number of different approaches for finding routes between networks are possible.
One useful way of categorizing these approaches is on the basis of the type of
information the gateways need to exchange in order to be able to find routes. Distance
vector algorithms are based on the exchange of only a small amount of information.
Each entity (gateway or host) that participates in the routing protocol is assumed to
keep information about all of the destinations within the system. Generally,
information about all entities connected to one network is summarized by a single
entry, which describes the route to all destinations on that network.
This summarization is possible because as far as IP is concerned, routing within a
network is invisible. Each entry in this routing database includes the next gateway to
which datagram’s destined for the entity should be sent. In addition, it includes a "metric" measuring the total distance to the entity. Distance is a somewhat generalized concept, which may cover the time delay in getting messages to the entity, the dollar cost of sending messages to it, etc. Distance vector algorithms get their name from the fact that it is possible to compute optimal routes when the only information exchanged is the list of these distances. Furthermore, information is only exchanged among entities that are adjacent, that is, entities that share a common network.

We said above that each entity keeps a routing database with one entry for every possible destination in the system. An actual implementation is likely to need to keep the following information about each destination:

- **Address**: in IP implementations of these algorithms, this will be the IP address of the host or network.
- **Gateway**: the first gateway along the route to the destination.
- **Interface**: the physical network which must be used to reach the first gateway.
- **Metric**: a number, indicating the distance (time delay, dollar cost, etc.) to the destination.
Chapter 5

Implementation
5.1 Tools and technologies used

5.1.1 Netbeans

Netbeans refers to both a platform for the development of applications for the network (using Java, JavaScript, PHP, Python, Ruby, Groovy, C, and C++), and an integrated development environment (IDE) developed using the NetBeans Platform.

The Netbeans Platform allows applications to be developed from a set of modular software components called modules. A module is a Java archive file that contains Java classes written to interact with the Netbeans Open APIs and a manifest file that identifies it as a module. Applications built on modules can be extended by adding new modules. Since modules can be developed independently, applications based on the Netbeans platform can be extended by third party developers.

The Netbeans Platform is a reusable framework for simplifying the development of other desktop applications. The platform offers services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of the platform are:
- User interface management (e.g. menus and toolbars)
- User settings management
- Storage management (saving and loading any kind of data)
- Window management
- Wizard framework (supports step-by-step dialogs)

The **NetBeans IDE** is an open-source integrated development environment written entirely in Java using the NetBeans Platform. NetBeans IDE supports development of all Java application types (Java SE, web, EJB and mobile applications) out of the box. Among other features are an Ant-based project system, version control and refactoring.

### 5.1.2 Sun Wireless Toolkit (WTK):

The Sun Java Wireless Toolkit (formerly known as Java 2 Platform, Micro Edition (J2ME) Wireless Toolkit) is a state-of-the-art toolbox for developing wireless applications that are based on J2ME’s Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP), and designed to run on cell phones, mainstream personal digital assistants, and other small mobile devices. The toolkit includes the emulation environments, performance optimization and tuning features, documentation, and examples that developers need to bring efficient and successful wireless applications to market quickly.

Projects can be configured for MIDP 1.0 or MIDP 2.0, with any combination of optional packages. The toolkit takes care of the details, which is very handy for building projects for specific stacks.

The J2ME Wireless Toolkit is a comprehensive set of tools for building MIDP applications. The toolkit can be used standalone, or incorporated into many popular integrated development environments (IDEs). The Sun J2ME Wireless Toolkit supports the development of Java applications that run on devices such as cellular phones, two-way pagers, and palmtops.
5.1.3 Bluetooth stacks (Widcomm and BlueZ):

**Widcomm**

WIDCOMM® software offers customers the most proven, easy porting, broad application support and the most proven Bluetooth stack available. WIDCOMM delivers a full range of wireless software systems and turnkey designs to enable PCs, PDAs, handsets, headsets, access points, and output devices with Bluetooth wireless technology. With the first Bluetooth SIG BQB certified product in the World, WIDCOMM continues to innovate the wireless communications market. Broadcom's Widcomm enables easy and cost effective integration of Bluetooth.
**BlueZ**

BlueZ is the official Linux Bluetooth protocol stack. It is an Open Source project distributed under GNU General Public License (GPL). BlueZ kernel is part of the official Linux kernel since version 2.4.6. BlueZ is the official Bluetooth stack for Linux and is used in Google’s Android OS. 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.

hidd is the Bluetooth human interface device (HID) daemon.

**5.1.4 MYSQL:**

MySQL is a relational database management system. MySQL is popular for web applications and acts as the database component. MySQL is written in C and C++. The SQL parser uses yacc and a home-brewed lexer, sql_lex.cc. MySQL works on many different system platforms, including AIX, BSDi, FreeBSD, HP-UX, i5/OS, Linux, Mac OS X, NetBSD etc. Libraries for accessing MySQL databases are available in all major programming languages with language-specific APIs. In addition, an ODBC interface called MyODBC allows additional programming languages that support the ODBC interface to communicate with a MySQL database, such as ASP or ColdFusion. The MySQL server and official libraries are mostly implemented in ANSI C/ANSI C++.
MySQL database is the world's most popular open source database because of its fast performance, high reliability, ease of use, and dramatic cost savings.

5.1.6 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. We have written the PC application of Bluetooth scatternet using the bluecove library for the blueZ stack in linux.

BlueCove provides an implementation of the JSR 82. BlueCove provides Java API for Bluetooth JSR 82. BlueCove rely upon already installed native Bluetooth stack that comes with you operating system or Bluetooth USB device.

Runtime Requirements:

- WIDCOMM (Broadcom) BTW Stack software version 1.4.2.10 SP5 or above
- BlueSoleil version 1.6.0, 2.3 or 3.2.2.8. Version 5.0.5 not supported.
- Microsoft Bluetooth stack (currently this means Windows XP SP2 or newer and Windows Mobile 2003 or newer)
- PowerPC- or Intel-based Mac OS X 10.4 (Bluetooth v1.2) or late (Since v2.0.2)
- Linux with BlueZ Bluetooth stack
- A Bluetooth device supported by the WIDCOMM, BlueSoleil or Microsoft bluetooth stack
- Java 1.1 or newer for the binary execution, Java 1.4 or newer to compile.
- Another Bluetooth device to communicate with.
Limitations

- Its support is available only on Windows WIDCOMM Stack, Linux BlueZ and Mac OS X Stack.
- Since Microsoft Bluetooth stack only supporting RFCOMM connections, BlueCove also supports only RFCOMM connections on this stack.
- If someone writes code to support another stack and/or operating system, it will be considered for inclusion.

5.1.7 J2SE : (Java Second Standard Edition)

Java has advanced through the years in both functionality and reach. J2SE is used primarily for writing applets and other Java-based applications.

The Java 2 Platform has three basic Editions: Micro, Standard, and Enterprise. They have increasing amounts of functionality and flexibility, with the Micro Edition being used for small-range applications and the Enterprise Edition being used for large, server-based functions. In the middle is the Standard Edition, or J2SE. J2SE has applications up and down the requirements ladder, filling needs for both individual and complicated users.

One of the primary uses of J2SE is the development of Java applications for individual computers. Another important functionality made possible by J2SE is JavaBeans. These are reusable applications that can be developed and assembled easily in order to create more sophisticated applications. Basically, they are the building blocks of personalized J2SE applications.

J2SE has the following features:

- Portability
- JDBC API for database access
- CORBA technology
- Java security.
J2SE consists of two components: Core Java and Desktop Java. Core Java provides back-end functionality, while Desktop Java provides GUI (Graphical User Interface) functionality. J2SE contains both the J2SE Development Kit (JDK) and the Java Runtime Environment (JRE).

J2SE 5.0 is a significant release including many new features and updates while preserving

5.1.8 JAVA ME.

Java ME is formerly known as J2ME. Java ME is the java platform for consumer and embedded devices such as mobile phones, pagers, personal organizers, television set-top boxes, automobile entertainment and navigation systems, Internet televisions, and Internet-enabled phones. Java ME is one of the three platform editions. The other two platform editions are Java Platform, Java Platform, Standard Edition (Java SE) for desktop computers and Java Card.

The Java ME platform brings the power and benefits of Java technology to consumer and embedded devices. The main goal of Java ME is to enable devices to dynamically download applications that leverage the native capabilities of each device.

Why JAVA Technology for Bluetooth Devices?

A standard API will let programmers write Bluetooth applications that work across many hardware platforms. To define this standard API, the Java language is the ideal choice. A java API enables applications to run on different types of hardware, operating systems, and classes of device. In addition to portability, the Java language provides several other benefits:
• Rapid development of applications because of better abstractions and high-level programming constructs provided by an object-oriented programming language.

• Ability to dynamically expand a program’s functionality during execution by loading classes at runtime.

• Classes file verification and security features that provide protection against malicious applications. These safeguards are required to customize devices by downloading applications.

• Standard with better user interfaces that support sophisticated user interaction.

• Large developer community. The number of people who program in the Java language is continuously growing. The developer talent needed for programming in the java language already exists, and there is no need to grow a developer community.

5.2 Source code listing

BtClient: This class provides many services like device discovery, service discovery, reply path information, handle error, send message etc.

• forwardMessage() : Forward current message received if destination device was not the current device

• newSearch() : Search s for new device in neighboring network to join Scatternet

• replyPathInfo() : Replies path information to new device which query s for path info

• reportError() : Reports error in case remote device moved out of range and removes device from routing table

• searchServices(): Starts service search on remote devices

• sendMessage() Sends message typed at user interface to ToAddress

• sendPathInfo() : Sends current device path info to all neighboring device
Bluetooth state: This class contains current state of Bluetooth device like is client running, is search completed etc.

- **addDev**(dev, devaddr) : Adds any new device discovery by server
- **getDevaddr()** Return device address of all device been discovered
- **getDevices()** : Returns current set of devices which have been discovered
- **putDevAddr**(Vector deviceaddr) : Stores address of current device list
- **putDevices**(Vector devices) : Stores current list of devices.
- **removeDev**(java.lang.String dev) : Removes address from current device list.

BtServer: This class handles creation and initialization of server thread

- **initializeServer()** : Initializes server components
- **startServer()** : Starts Bluetooth server

ClientHandler: This class handles multiple clients and check header of received message.

- **addNewDevice()** : Add a new device which requested at served.
- **checkHeader**(java.lang.String s) : Check s header of message and processes client request accordingly
- **checkRecieved**(java.lang.String s) : Retrieves contents of message been received
- **extractInfo**(java.lang.String s) : Extract s routing information from message being received
- **removeDevice**(java.lang.String s) : Removes un available remote device from list
- **serviceClient**(javax.microedition.io.StreamConnection con) : Service s client request
Chapter 5

Testing
6.1 Testing Scatternet Application.

**Software testing** is the process used to assess the quality of computer software. Software testing is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding software bugs. Quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; testing furnishes a criticism or comparison that compares the state and behavior of the product against a specification. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (S.Q.A.), which encompasses all business process areas, not just testing.

Scatternet Application cannot be separated tested using wireless tool kit emulator because there was no provision to set Bluetooth range so as to form a Scatternet thus resulting in direct path between every device. In order to this application separate module was added which gives provision to make required devices invisible so that Scatternet can be simulated.

The snapshot shown here shows the initial screen of the mobile simulation Scatternet application. The only difference you might notice here from the other application is the extra option of a virtual Scatternet. This option is to simulate a Scatternet without using actual devices. This can be used to demonstrate the Scatternet over a large number of hops. Figure 3.1 below shows the snapshot of three emulators running the application and on which the virtual Scatternet option has been selected. On each screen you get a list of devices.
visible to it. Here you may notice that the first device has the address of the second and
the second device has the address of the first. The third device is visible to neither but it
detects the other two and then sends its information to it. Figure 3.2 shows the routing
table.

![Figure 3.1](image1)

![Figure 3.2](image2)

You can notice the hop count as being two in the case of the third device when it wants to
send it to the first device.
Conclusion

Scatternet have the potential to bring the interconnectivity of the Internet to the physical world through wireless devices. A number of companies have attempted to launch social networking and dating services that leverage early Scatternet implementations. Scatternets can also be used to enable ad-hoc communication and interaction between autonomous robots and other devices.

Future Enhancements

1. The Scatternet application currently runs successfully on s40 devices and Linux systems with BlueZ Bluetooth stack. The aim is to successfully run it on almost all mobile devices with a decent processing capability and on prominent Bluetooth stacks such as bluesoleil.

2. The application will be improved to support large hops. Its in this regard that we look for a more efficient algorithm.

3. With the advent of Bluetooth 3.0 in the near future, the application shall be improved taking this into consideration. Bluetooth 3.0 promises to have a much larger bandwidth and an inherent support for Wi-Fi making Scatternets all the more useful and viable.

Applicability

Let us now take a look at certain scenarios and situations where a Scatternet will come into use.

1. **Scenario – Office**: By installing a Bluetooth network in your office you can do away with the complex and tedious task of networking between the computing
devices, yet have the power of connected devices. No longer would you be bound to fixed locations where you can connect to the network. Each Bluetooth device could be connected to 200 other devices making the connection of every device with every other possible. Since it supports both point to point and point to multipoint it will virtually make the maximum number of simultaneously linked devices unlimited.

2. **Scenario – Home:** Upon arriving at your home, the door automatically unlocks for you, the entry way lights come on, and the heat is adjusted to your pre-set preferences.

3. **Scenario – Recreation:** If your camera is Bluetooth enabled, you can transfer your snapshots immediately into a storage device without any hassles of wires and connecting them. You can also with the extended range of a Bluetooth network make listening to your music from your iPod or an mp3 player just a bit more fun!
References

3. ETH Zurich
4. BTnode project.