A New Two-Phase Scatternet Formation Algorithm
for Bluetooth Wireless Personal Area Networks

by

CHU ZHANG

B.Sc., Wuhan University, China 1996

A THESIS SUBMITTED IN FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

December 2003

© Chu Zhang, 2003
In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Chu Zhang

Name of Author (please print)

23/12/2003

Date (dd/mm/yyyy)

Title of Thesis: A New Two Phase Scatternet Formation (TPSF) Algorithm for Bluetooth Wireless Personal Area Networks

Degree: Master of Applied Science

Year: 2004

Department of Electrical and Computer Engineering

The University of British Columbia

Vancouver, BC Canada
Abstract

A Bluetooth multi-hop personal area network can be formed by interconnecting one or more piconets into a scatternet. A Bluetooth scatternet is an ad hoc network in which the devices move randomly and organize themselves. The scatternet is attractive because it can extend the Bluetooth radio range and improve the network capacity. The current Bluetooth specification [1] only defines the scatternet but does not address how the scatternet is formed. To reduce the high load on the master nodes and bridge nodes, a two-phase scatternet formation (TPSF) [8] algorithm has been proposed in which a control scatternet is created for the control traffic in the first phase and an on-demand scatternet is created for the data traffic in the second phase. In TPSF, route information for the on-demand scatternet on each node is discovered only when the node initially accesses the network. The original TPSF does not consider the support of mobility. In this thesis, we propose a new scheme which is called TPSF+ for the on-demand scatternet formation in the second phase of TPSF. In TPSF+, route information is discovered when a communication session is required between the two nodes. Consequently, the on-demand scatternet can be formed with much higher success ratio when the slaves randomly move around the master after accessing the network. We also propose to use PM_ADDR (Parked Member Address) instead of BD_ADDR (Bluetooth Device Address) during route discovery in order to reduce the time of the route discovery process. Furthermore, to reduce the hop distance of the on-demand scatternet, we limit the number of hops in each piconet in the control scatternet. Based on the simulation results, we show that our scheme can improve network performance greatly in terms of aggregate throughput and end-to-end delay even with the consideration of packet collisions. With the slaves randomly moving around the master, TPSF+ achieves much better performance in terms of a higher successful path connection ratio.
# TABLE OF CONTENTS

Abstract ........................................................................................................... ii

List of Tables ................................................................................................... vii

List of Figures ................................................................................................. viii

Acknowledgements .......................................................................................... x

Chapter 1 Introduction ..................................................................................... 1

  1.1 Motivations and Contributions ................................................................. 1

  1.2 Organization .............................................................................................. 4

Chapter 2 Overview and Related Work ........................................................... 5

  2.1 Overview .................................................................................................... 5

    2.1.1 Radio Link Layer ................................................................................. 7

    2.1.2 Baseband Layer ................................................................................. 7

      2.1.2.1 Physical Channel ......................................................................... 8

      2.1.2.2 Physical Links ............................................................................. 9

      2.1.2.3 Error Detection and Correction ................................................. 10

      2.1.2.4 Packet Format ........................................................................... 10

      2.1.2.5 Addressing ............................................................................... 12

      2.1.2.6 Connection Establishment ......................................................... 13

      2.1.2.7 Connection State .................................................................... 16

    2.1.3 Link Management Protocol (LMP) Layer ........................................ 18

    2.1.4 Logical Link Control and Adaptation Protocol (L2CAP) Layer ....... 18

    2.1.5 Challenges in Bluetooth Scatternet Formation ............................... 18
2.2 Related Work

2.2.1 Bluetooth Topology Construction Protocol (BTCP)

2.2.2 Bluetrees

2.2.3 Bluenet

2.2.4 Scatternet Formation Algorithm Proposed by Law and Siu

2.2.5 Tree Scatternet Formation (TSF)

2.2.6 Loop Scatternet Formation (LSF)

2.2.7 Bluestar

2.2.8 Various Topologies Proposed by Guerin et al.

2.2.9 Scatternet Route

2.2.10 Two-Phase Scatternet Formation (TPSF) Algorithm

2.3 Comparisons of Previous Scatternet Formation Algorithms

2.4 Problem Definition

Chapter 3  A New Two-Phase Scatternet Formation (TPSF+) Algorithm

3.1 Control Scatternet Formation

3.2 On-demand Scatternet Formation

3.2.1 Masters' Route Discovery Procedure

3.2.2 On-demand Scatternet Route Discovery and Connection

3.2.2.1 PM_ADDR Scheme

3.2.2.2 Maximum Intra-piconet Distance

3.2.2.3 On-demand Scatternet Formation Procedure

3.3 Packet Formats

3.3.1 Masters' Route Request (MRREQ) Packet and Masters' Route Re-
Bibliography

74
List of Tables

Table 2.1  ACL Packets ................................................................. 11
Table 2.2  A Comparison of Proactive Scatternet Formation Algorithms ............. 31
Table 3.1  Masters' Route Table ....................................................... 40
Table 3.2  Mapping Table in Piconet M1 ........................................... 40
Table 3.3  Mapping Table in Piconet M2 ........................................... 40
Table 3.4  Mapping Table in Piconet M3 ........................................... 40
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2.1</td>
<td>Bluetooth Protocol Stack</td>
<td>6</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Packet Format</td>
<td>12</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Format of BD_ADDR</td>
<td>12</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>Inquiry Procedure</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>Paging Procedure</td>
<td>15</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>Sniff Mode</td>
<td>17</td>
</tr>
<tr>
<td>Figure 2.7</td>
<td>Hold Mode</td>
<td>17</td>
</tr>
<tr>
<td>Figure 2.8</td>
<td>Park Mode</td>
<td>17</td>
</tr>
<tr>
<td>Figure 2.9</td>
<td>BTCP Establishment Procedure</td>
<td>21</td>
</tr>
<tr>
<td>Figure 2.10</td>
<td>A Bluetrees Scatternet</td>
<td>22</td>
</tr>
<tr>
<td>Figure 2.11</td>
<td>A Bluenet Scatternet</td>
<td>23</td>
</tr>
<tr>
<td>Figure 2.12</td>
<td>An Example of a New Scatternet Formation</td>
<td>24</td>
</tr>
<tr>
<td>Figure 2.13</td>
<td>A Scatternet with 20 Nodes Created by TSF [6]</td>
<td>25</td>
</tr>
<tr>
<td>Figure 2.14</td>
<td>A Loop Scatternet</td>
<td>26</td>
</tr>
<tr>
<td>Figure 2.15</td>
<td>A Bluestar Scatternet</td>
<td>27</td>
</tr>
<tr>
<td>Figure 2.16</td>
<td>A Scatternet Route</td>
<td>28</td>
</tr>
<tr>
<td>Figure 2.17</td>
<td>A Two Phase Scatternet Formation Protocol [14]</td>
<td>30</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Masters' Route Discovery Procedure</td>
<td>39</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>On-demand Scatternet Route Discovery Procedure</td>
<td>45</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>On-demand Scatternet Route through the Master and Bridge Nodes</td>
<td>46</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>MRREQ Packet Format</td>
<td>47</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>MRREP Packet Format</td>
<td>49</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>OSRREQ Packet Format</td>
<td>50</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Node Structure</td>
<td>53</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Multiple CIDs in L2CAP Layer</td>
<td>55</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Interference Model</td>
<td>56</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>A Control Scatternet which consists of a Single Piconet. The Corresponding One-hop On-demand Scatternet</td>
<td>57</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>A Control Scatternet which consists of Two Piconets. An On-demand Scatternet with Two Hops</td>
<td>58</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>Aggregate Throughput of DH3 Packet</td>
<td>60</td>
</tr>
<tr>
<td>Figure 4.7</td>
<td>End-to-End Delay of DH3 Packet</td>
<td>60</td>
</tr>
<tr>
<td>Figure 4.8</td>
<td>Aggregate Throughput of DH5 Packet</td>
<td>61</td>
</tr>
<tr>
<td>Figure 4.9</td>
<td>End-to-End Delay of DH5 Packet</td>
<td>61</td>
</tr>
<tr>
<td>Figure 4.10</td>
<td>TCP Throughput Performance</td>
<td>63</td>
</tr>
<tr>
<td>Figure 4.11</td>
<td>TCP End-to-end Delay Performance</td>
<td>63</td>
</tr>
<tr>
<td>Figure 4.12</td>
<td>On-demand Scatternet Connection Delay</td>
<td>64</td>
</tr>
<tr>
<td>Figure 4.13</td>
<td>Successful Path Connection Ratio among Slaves</td>
<td>64</td>
</tr>
<tr>
<td>Figure 4.14</td>
<td>Average Hop Distance with 10 Slaves per Piconet</td>
<td>66</td>
</tr>
<tr>
<td>Figure 4.15</td>
<td>Average Hop Distance with 20 Slaves per Piconet</td>
<td>66</td>
</tr>
<tr>
<td>Figure 4.16</td>
<td>Average Hop Distance with 30 Slaves per Piconet</td>
<td>67</td>
</tr>
<tr>
<td>Figure 4.17</td>
<td>Successful Path Connection Ratio</td>
<td>67</td>
</tr>
</tbody>
</table>
Acknowledgements

First of all, I would like to thank my supervisor, Professor Victor Leung. This thesis would not have materialized without his constant supervision. It has been an incredible experience and an invaluable opportunity to study under his guidance. I am also grateful to him for allowing me to present our work at various meetings in our department.

I would also like to thank my co-supervisor Professor Vincent Wong. He has always given me mental support and thoughtful advice. He has devoted a lot of time and effort, helping me improve my writing, presentation and technical skills with weekly meetings and showing me how to effectively solve problems and arrange my schedule.

During my research, I had the opportunity to work with Mr. Yoji Kawamoto who was a former visitor scholar in our communications research group and who is now working at Sony Corporation in Japan. I appreciate his insightful comments on my work. And I was able to expand his work. He gave me perspective on my thesis.

I am grateful to Mr. Joohan Song for helping me clarify various issues and giving me some helpful suggestions on ns-2 simulations.

I also would like to thank all the members in our communications research group for their help and support, especially Mary Ma and Fei Yu. They also gave me some suggestions on my simulation work.

And finally, to my family, I am forever indebted to you. I should give my sincere thanks to my parents, Wenyi Zhang and Renfang Dou for their spiritual support and endless encouragement. To my husband, Qiang Hou, thank you for your understanding, constant love and patience throughout my graduate studies at UBC.
Chapter 1 Introduction

In the last few years, many wireless connectivity standards/technologies have been developed. Among them, there are IrDA [1], HomeRF [2], IEEE 802.11 [3] and Bluetooth [4]. These technologies enable a wide range of electronic devices to be easily and simply connected, without requiring cables. With the advantage of low power consumption, no line-of-sight requirement and the use of the unlicensed frequency band, Bluetooth has become the promising standard among those short-range wireless technologies.

The Bluetooth wireless technology uses the globally available 2.4 GHz frequency band, which ensures wide compatibility in the communication world. Because Bluetooth utilizes a radio-based link, it does not require a line-of-sight connection. For example, using Bluetooth, a desktop personal computer (PC) can send information to a printer in a neighboring room, or one may use a mobile phone to control a home electronic devices within the radio range. Bluetooth Personal Area Network (PAN) is a self-organizing network. It does not require much configurations and installations like a Wireless Local Area Network (LAN). Whenever any Bluetooth-enabled devices are within a 10-meter range, they can instantly exchange addresses and clock information and then establish a Bluetooth PAN.

Bluetooth has already become a global de facto standard for short-range wireless connectivity. With the potential uses as mentioned above, there is no doubt that Bluetooth will become one of the most widely adopted wireless technologies.

1.1 Motivation and Contributions

Since the number of electronic devices is increasing quickly, more research is focused on how to connect them. Bluetooth creates a PAN in which all the Bluetooth-enabled devices are automatically connected without wired lines. You can imagine yourself waiting in an
airport. Suppose the departure time of your plane is delayed and you have extra time to do some work. You can sit down, use your personal digital assistant (PDA) to download some documents from your laptop PC, and then browse the documents on your PDA or you can use your PDA to browse the news or email via your laptop PC which is connected with the Internet. You can also use the headset to listen to MP3 music stored on your laptop without any cable. Although Bluetooth was originally created primarily for wireless phones and headsets, now there are all sorts of Bluetooth-enabled devices including PDAs, printers, keyboards and mice.

Bluetooth has specified the *piconet* in which one master can have at most seven active slaves. In future, it will extend to a large-scale structure called *scatternet* where the piconets are interconnected. As the scatternet can extend the Bluetooth radio range and enlarge the network capacity, recent research on scatternet formation protocols has received increased attention. Bluetooth radio can provide short-range wireless communications by one hop. Using scatternet, Bluetooth can implement multi-hop communications so the radio range can be greatly extended. For example, suppose you wish to connect your PDA to a PC which is further than one radio range. Using Bluetooth technology, you can connect your PDA with the PC through any other intermediate Bluetooth-enabled devices instead of walking closer to the PC. In addition, by forming the scatternet, more network capacity can be obtained. The total capacity of a Bluetooth piconet is only about 1 Mbps. However there is more than one piconet in the Bluetooth scatternet and each piconet has the capacity of 1 Mbps. Therefore, the aggregate network capacity can be increased to a multiple of 1 Mbps.

Although the current Bluetooth specification [4] defines the scatternet, the scatternet formation issue is not fully resolved. Like the Bluetooth piconet, Bluetooth scatternet
operates in an ad hoc fashion. Bluetooth scatternet targets PANs which connect all kinds
of electronic devices. Furthermore, mobility of the nodes requires the scatternet to support
dynamic operations, which make scatternet formation more challenging. The mobility
includes two scenarios: (1) Node joins or leaves the network. (2) Node randomly moves
within the scatternet.

The new algorithm proposed in this thesis is mainly aimed at supporting node
mobility in scatternet formation. We propose a new scatternet formation algorithm, which
is the extension of TPSF (Two-Phase Scatternet Formation) algorithm [14]. The new
algorithm is called TPSF+, which is a combination of proactive and reactive algorithms. In
TPSF+, the control scatternet is formed proactively as the first phase in TPSF [14],
whereas the on-demand scatternet route information is obtained reactively when the
source node wants to communicate with the destination node in the second phase. The
main contributions of this thesis are as follows:

• TPSF+ algorithm can create an on-demand scatternet with a high successful path
  connection ratio in a dynamic environment. TPSF+ can allow the slaves to
  randomly move around the masters in the control scatternet without the
  requirement of additional update messages. In order to show this improvement, a
  mobility model is built for the performance comparison with TPSF. In a dynamic
  environment in which the slaves randomly move around the master of the control
  scatternet, we are able to demonstrate that the successful path connection ratio for
  the on-demand scatternet is much higher using TPSF+ than using TPSF.

• To minimize the average hop distance of the on-demand scatternet, the maximum
  intra-piconet distance scheme is proposed. The maximum intra-piconet distance is
  defined as the maximum number of hops the on-demand scatternet route request
packet is allowed to traverse in a single piconet within the control scatternet. The simulation results show that TPSF+ can achieve a smaller average hop distance for the on-demand scatternet.

- TPSF+ has the advantage of involving a small number of nodes participating in the on-demand scatternet route discovery procedure so as to avoid unnecessary route discoveries. The on-demand scatternet route discovery is limited to several piconets instead of all the piconets within the control scatternet.

- In TPSF+, we propose that the PM_ADDR (Park Mode Member Address) be applied to the on-demand scatternet route discovery instead of BD_ADDR (Bluetooth Device Address). Our results show that the use of PM_ADDR can make the control packets more efficient.

- A packet collision model is built for performance study of the resulting on-demand scatternet. Based on the packet collision model, the performance of TPSF+ is compared with another scheme - Bluetooth Topology Construction Protocol (BTCP) [5], in terms of aggregate throughput and end-to-end delay.

1.2 Organization

This thesis is organized as follows. In Chapter 2, we present an overview of Bluetooth technology and provide a literature survey of Bluetooth scatternet formation algorithms. In Chapter 3, we propose a new Two-Phase Scatternet Formation+ (TPSF+) algorithm based on the TPSF algorithm. Chapter 4 describes the simulation model and presents the results. Conclusions and future work are given in Chapter 5.
Chapter 2  Overview and Related Work

In this Chapter, an overview of the Bluetooth technology including its history and protocol architecture is given in Section 2.1. After introducing the basics of Bluetooth, the related work of the scatternet formation algorithms in the literature is described in Section 2.2. Afterwards, a qualitative comparison of previous scatternet formation algorithms is given in Section 2.3. Finally, the problem definition is stated in Section 2.4.

2.1 Overview

Bluetooth was initially developed by Ericsson as a short-range communication technology to replace cables in 1994. Bluetooth is named after the Danish king named Harald Blantand, which means “Bluetooth” in English. The name is appropriate because just as King Harald Blantand united Denmark and Norway the goal of Bluetooth technology is to unit all kinds of electronic devices by connecting them wirelessly.

Later Bluetooth was developed by the Bluetooth Special Interests Group (SIG) which was founded in 1998. Nowadays many companies world-wide, including Nokia, Motorola, Qualcomm, Compaq, Dell, 3Com, and Lucent, have joined the Bluetooth revolution and became the members of Bluetooth SIG, helping to expand the Bluetooth specification.

In the Bluetooth network, all devices are peer units, identified by a unique 48-bit address. The device initializing the connection is assigned as a master. The connected device is assigned as a slave. By forming a piconet, a master can control up to 7 active slaves. All the devices in the same piconet synchronize to a frequency hopping channel, which is determined by the address of the master device. A master device controls a piconet based on a time-division multiplexing scheme. If multiple piconets are connected, some Bluetooth devices may participate in more than one piconet on a time-division basis.
Chapter 2 Overview and Related Work

Such devices are served as bridge nodes and the connected piconets form a scatternet.

The Bluetooth protocol architecture is constructed in a layered approach similar to the Open System Interconnection (OSI) model. It is divided into five layers, namely the radio link layer, baseband layer, Link Management Protocol (LMP) layer, Logical Link Control and Adaptation Protocol (L2CAP) layer, as shown in Figure 2.1. The radio link layer defines the output power, the modulation technique, and the frequency hopping schemes used in Bluetooth. The radio link layer is analogous to the physical layer of the OSI model. Based on the radio link layer, the baseband layer describes how the connection is set up among different devices. The LMP layer is responsible for the link configuration, attachment, detachment, and power control. The L2CAP layer takes data from higher protocol layers and application and supports higher level protocol multiplexing, packet segmentation and reassembly, and the conveying of quality of service information. The baseband layer, LMP layer, and L2CAP layer correspond to the data link layer of OSI model. Transmission Control Protocol/Internet Protocol (TCP/IP) or User Datagram
Protocol (UDP) can run directly on the L2CAP layer.

2.1.1 Radio Link Layer

The Bluetooth radio link layer is the lowest defined layer in the Bluetooth specification [4]. It defines the requirements of the Bluetooth transceiver device operating in the 2.4 GHz Industrial-Scientific-Medical (ISM) band.

**Power Class:** Three power classes are defined in the Bluetooth devices. Power Class 1 is designed for long range (∼100 m) devices, with a maximum output power of 20 dBm. Power Class 2 is for ordinary range (∼10 m) devices, with a maximum output power of 4 dBm. Power Class 3 is for short range (∼10 cm) devices, with a maximum output power of 0 dBm.

The Bluetooth radio interface is based on a nominal antenna power of 0 dBm. Each device can vary its transmitted power by measuring Radio Signal Strength Indicator (RSSI) which is an optional function of Bluetooth. Equipment with power control capability can optimize the output power in a link by using LMP commands.

**Frequency band:** Bluetooth specifically operates in the license-free 2.4 - 2.4835 GHz ISM band. To combat the multi-path or interference effects, Bluetooth uses the frequency hopping spread spectrum scheme in which there are 79 hops, spanning from 2.402 GHz to 2.480 GHz. In some other countries such as France and Spain, a reduced 23-hop system is used.

2.1.2 Baseband Layer

The baseband layer is built on top of the radio link layer in the Bluetooth protocol. Operated as a link controller, the baseband layer works with LMP layer for implementing link level routines such as link connection and power control. The baseband layer also
implements Bluetooth device discovery in the radio range by inquiry, performs connection setup by paging, manages asynchronous and synchronous links and handles packets.

2.1.2.1 Physical Channel

On the baseband layer, the channel is divided into time slots where each slot corresponds to a Radio Frequency (RF) hop frequency. The frequency hopping scheme in Bluetooth is based on time-slot-by-time-slot. Every time slot takes up 625 μs. Thus the hopping rate is 1600 hops/sec. The spread spectrum frequency hopping works in a way that the Bluetooth device hops to different frequencies 1,600 times per second, switching between each 1 MHz band. Using the hop sequence, the Bluetooth device can transmit on one hop and then listen on another hop alternatively.

The Bluetooth channel is identified by a pseudo-random hopping sequence hopping through the 79 (or 23) RF channels. Bluetooth provides point-to-point or point-to-multi-point connection by forming a piconet. The hopping sequence is determined by the BD_ADDR (Bluetooth device address) of the master and the phase in the hopping sequence is determined by the clock of the master. The hopping sequence is unique for each piconet.

There are ten types of hopping sequences defined in Bluetooth, five types in the 23-hop system and the other five types in the 79-hop system. The five types of hopping sequences in the 79-hop system are the inquiry sequence, inquiry response sequence, page sequence, page response sequence, and the channel hopping sequence.

The inquiry sequence and inquiry response sequence are derived from a General Inquiry Access Code (GIAC: 0x9E8B33), which is used to discover all devices in the radio range. The page sequence and page response sequence are derived from the address
of the slave device. The channel hopping sequence is derived from the address of the master device in the piconet.

### 2.1.2.2 Physical Links

The Baseband layer supports two types of physical links: Synchronous Connection-Oriented (SCO) links and Asynchronous Connection-Less (ACL) links. The SCO link mainly supports real-time voice traffic using reserved bandwidth while the ACL link supports best effort traffic.

The SCO link is a symmetric point-to-point link between a master and a single active slave in the piconet. Once the connection is established, both the master and slave units may send SCO packets over reserved intervals; therefore, the SCO link is also called the circuit switched type link. SCO packets are never retransmitted. SCO packets are mainly used for 64 kb/sec speech transmission. As in the Bluetooth specification [4], the master can support up to three simultaneous SCO links while the slaves can support two or three SCO links.

The ACL link is a point-to-multipoint link between the master and its slaves. The ACL link supports both symmetric and asymmetric traffic. The master can establish an ACL link on a per-slot basis with any slave. The master controls the link bandwidth and decides the communication with each slave by polling the slave. The slave must be polled by its master before it can transmit data to the master. Only one single ACL link can exist between the master and each slave. In case of error, packet retransmission is required for most ACL packets.

### 2.1.2.3 Error Detection and Correction

There are two error-correction schemes defined by Bluetooth baseband controllers:
Chapter 2 Overview and Related Work

Forward Error Correction (FEC) code and Automatic Repeat Request (ARQ) scheme.

The purpose of applying FEC to the data payload is to reduce the number of retransmissions. The packet header is protected by a 1/3 rate FEC all the time because it contains significant link information and should be able to be corrected in case of bit errors. The payload can be optionally protected by a 2/3 rate FEC. FEC prevents some unnecessary retransmissions but it increases the overhead at the same time. In a reasonably error-free environment, FEC increases unnecessary overhead and reduces the throughput.

An unnumbered ARQ scheme is used in which the packet is transmitted in one slot and directly acknowledged by the recipient in the next slot. The packet can be sent only when the last packet is acknowledged. It works in the same way as the Stop-and-Wait ARQ scheme.

2.1.2.4 Packet Format

There are 13 different packet types defined in the baseband layer, 3 types in SCO links and 10 types in ACL links. These packets can be used to compose higher level Protocol Data Units (PDUs) by higher layers. For ACL links, the packets are ID, POLL, NULL, FHS, DM1, DH1, DM3, DH3, DM5, DH5. The size of an ID packet is 68 bits. It is normally used in paging, inquiry and response procedures. A POLL packet is 126 bits long, consisting of the CAC (Channel Access Code) and packet header only. The POLL packet is sent from a master to a slave for scheduling and the slave must respond with a packet after receiving the POLL packet. The NULL packet is similar to the POLL packet, except it does not require confirmation from the receiver. It is used to return link information to the source. FHS (Frequency Hopping Synchronization) packet is a control packet carrying the BD_ADDR and the clock of the sender. It contains 144 information bits and a 16-bit
Table 2.1 ACL Packets

<table>
<thead>
<tr>
<th>Packet Types</th>
<th>Timeslots</th>
<th>Payload Length (Bytes)</th>
<th>FEC</th>
<th>Symmetric Max. Data Rate (kb/sec)</th>
<th>Asymmetric Max. Data Rate (kb/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Forward</td>
<td>Reverse</td>
</tr>
<tr>
<td>DH1</td>
<td>1</td>
<td>0-27</td>
<td>no</td>
<td>172.8</td>
<td>172.3</td>
</tr>
<tr>
<td>DH3</td>
<td>3</td>
<td>0-183</td>
<td>no</td>
<td>390.4</td>
<td>585.6</td>
</tr>
<tr>
<td>DH5</td>
<td>5</td>
<td>0-339</td>
<td>no</td>
<td>433.9</td>
<td>723.2</td>
</tr>
<tr>
<td>DM1</td>
<td>1</td>
<td>0-17</td>
<td>yes</td>
<td>108.8</td>
<td>108.8</td>
</tr>
<tr>
<td>DM3</td>
<td>3</td>
<td>0-121</td>
<td>yes</td>
<td>258.1</td>
<td>387.2</td>
</tr>
<tr>
<td>DM5</td>
<td>5</td>
<td>0-224</td>
<td>yes</td>
<td>286.7</td>
<td>477.8</td>
</tr>
</tbody>
</table>

Cyclic Redundancy Check (CRC) code. The payload of the FHS packet is coded with a rate 2/3 FEC which makes the total payload length to be 240 bits. The FHS packet covers only a single time slot. DM packet represents "Data packet - Medium Rate" and is encoded by 2/3 FEC. DM1, DM3, and DM5 are the 1-slot, 3-slot, and 5-slot ACL link packets for medium rate respectively. DH packet represents "Data packet - High Rate". DH1, DH3, and DH5 are the 1-slot, 3-slot, and 5-slot ACL link packets for high rate respectively.

The general packet format is shown in Figure 2.2. Each packet consists of 3 fields: the access code (72 bits), the header (54 bits), and the payload (0-2745 bits).

Access Code: Access code is used for timing and synchronization, offset compensation, paging, and inquiry. There are three different types of access code: Channel Access Code (CAC) which identifies a unique piconet; Device Access Code (DAC) which is used for paging and its responses; and Inquiry Access Code (IAC), which is used for inquiry purpose.
Chapter 2 Overview and Related Work

2.1.2.5 Addressing

Three kinds of addresses defined in the Bluetooth specification are used in this thesis. They are as follows:

**Bluetooth Device Address (BD_ADDR):** Each Bluetooth transceiver is assigned a globally unique IEEE Medium Access Control (MAC) address which is 48-bit long. The format of the address is shown in Figure 2.3, which contains LAP (Lower Address Part), UAP (Upper Address Part), and NAP (Non-significant Address Part).

**Active Member Address (AM_ADDR):** In the piconet, the master assigns each of its active slaves a 3-bit active member address (AM_ADDR). The master has no AM_ADDR. AM_ADDR is only valid where the slave is either in the active, hold or sniff state. A node in park mode gives up its AM_ADDR. The all-zeros AM_ADDR is reserved as a broadcast address so that each master can have at most seven slaves. AM_ADDR is
carried in the data packet header or in the payload of FHS packet.

**Parked Member Address (PM_ADDR):** A slave in park mode is assigned a PM_ADDR. The PM_ADDR is 8-bit long. The slave in park mode can also be identified by its BD_ADDR; thus, the number of parked slaves has no limitations if the slave in park mode is identified by BD_ADDR and PM_ADDR is all-zero bits.

### 2.1.2.6 Connection Establishment

In order to set up the connection, the access procedure should be used. The access procedure in Bluetooth includes the inquiry procedure and paging procedure. The inquiry procedure is used to discover other devices in the radio range and the paging procedure is used to synchronize and set up the connection between the two devices.

**Inquiry procedure:** The inquiry procedure is used to gather the information of other Bluetooth devices within the radio range.

The inquiry procedure is shown in Figure 2.4. In the inquiry procedure, the device that wants to collect the information of other Bluetooth devices is called a sender and the device that wants itself to be discovered is called a receiver. The sender goes into the inquiry state and the receiver goes into inquiry scan state. The inquiry procedure is as follows:

First, the sender broadcasts the inquiry ID packet to the receivers around it and listens to the response. The receiver scans the ID packets sent by the sender. They all use the same 32 inquiry hop frequencies generated by the GIAC. The phase in the sequence is determined by the native clock of the bluetooth device. The sender hops at a faster rate than the receiver so the receiver can catch one of the ID packets [5].

Upon receiving the inquiry ID packet, the receiver returns to the state before the
The inquiry scan state and backs off for a random period that is determined by a random number distributed between 0 and 1023 in the unit of the timeslot.

After the random period, the receiver wakes up and goes into the inquiry response state and then it starts to scan again.

Finally, when the receiver receives the inquiry ID packet in the inquiry response state, it sends the FHS packet, which contains the receiver's address and clock value to the sender in the odd timeslot. At this time, the sender obtains the BD_ADDR and clock value of the receiver which are carried by the received FHS packet and then the inquiry procedure is finished.
Paging procedure: The paging procedure is used to set up the connection between two devices. In this procedure, the device that initiates the paging procedure will become the master after connection, and the device that enters the page scan will become the slave after connection.

The paging procedure is shown in Figure 2.5. In this procedure, the master sends an ID packet on the frequency to which the receiver is listening to. The frequency is determined by BD_ADDR and the clock information of the slave collected in the inquiry procedure. Then the slave replies with an ID packet. Upon receiving the ID packet from the slave, the master then sends an FHS packet, which contains the master’s BD_ADDR and clock value. In this way the slave obtains the BD_ADDR and clock value of its master node and then it can synchronize and hop with the master node. After the paging procedure, the master finishes creating the point-to-point connection with the slave on the baseband layer.
2.1.2.7 Connection State

After successfully performing the inquiry and paging procedure, the two devices are in the connection states, one as the master and the other as the slave. In the connection state, the slave node has four modes, namely active, hold, sniff, and park mode. The hold, sniff, and park mode are also called low-power modes.

**Active mode:** The node in active mode listens to the channel all the time. The master schedules the transmission to or from different slaves based on traffic demands. The slave in active mode is assigned by AM_ADDR. It listens to each master-to-slave slots and decides whether the packet received should be accepted depending on the AM_ADDR in the received packet.

**Sniff mode:** The slave wakes up and listens to the channel during each interval $T_{\text{sniff}}$ for $N_{\text{sniff attempt}}$ slots [4]. $N_{\text{sniff attempt}}$ is the number of slots that the slave can listen to during the interval $T_{\text{sniff}}$. If the slave receives a data packet, it will continue listening for subsequent $N_{\text{sniff timeout}}$. Otherwise, the slave will go into sleep state. $N_{\text{sniff timeout}}$ determines how many additional timeslots the slave must listen if it receives a packet. The sniff mode is shown in Figure 2.6.

**Hold mode:** In hold mode, the master and slaves agree on the same time period during which the slave does not participate in the channels temporarily. During the hold period, the slave can do other things such as scanning, paging, and inquiring. The bridge node can use the hold mode to switch between different masters. The hold mode is shown in Figure 2.7.

**Park mode:** In park mode, the slave wakes up only at an interval for synchronization and sleeps all the other times. This interval is agreed between the master node and the slave node before the slave node goes into the park mode. The slave will give
Chapter 2 Overview and Related Work

Master-to-slave

Master

Slave

Listen

$N_{\text{sniff attempt}}$

$T_{\text{sniff}}$

Figure 2.6 Sniff Mode

Hold Duration

Master

Hold Request

Hold Response

Slave

Figure 2.7 Hold Mode

Park Time

Master

Park Request

Park Response

Slave

Figure 2.8 Park Mode
up its AM_ADDR and use its PM_ADDR or BD_ADDR after it goes into the park mode. Using the park mode, the number of slaves in the piconet can be more than seven. The park mode is shown in Figure 2.8.

2.1.3 Link Management Protocol (LMP) Layer
The LMP layer carries out the link setup, security setup, and link configuration for both voice and data. The LMP also manages the different low-power modes defined in the Baseband Layer: sniff, hold and park. LMP commands are always sent as single-slot packets and the payload header is therefore one byte.

2.1.4 Logical Link Control and Adaptation Protocol (L2CAP) Layer
The L2CAP layer is on the upper layer of the LMP and resides in the data link layer. L2CAP provides the protocol multiplexing capability, segmentation and reassembly operation to the upper layer protocols for ACL. No support for SCO links is specified in L2CAP layer. For SCO links, the application can use the baseband directly.

2.1.5 Challenges in Bluetooth Scatternet Formation
As mentioned earlier, the Bluetooth scatternet is a self configuring network that needs to support node mobility. There are still many issues on how to organize the nodes into an operational network. Based on the Bluetooth specification [4], the problems are as follows:

(1) How does each node choose the role in the final scatternet?
(2) How many piconets should be in the final scatternet?
(3) How many piconets should a bridge node belong to?
(4) How many bridges should a piconet have?
(5) How many slaves should the master control in one piconet?
(6) Which mode should the slave use in the final scatternet?
Most of above questions are mentioned in [12]. Question (1) raises the issue of how to decide which node should be the master, slave, or bridge in the scatternet.

The degree of the master node is the number of the slaves that are connected to the master node. The degree of the bridge node is the number of the piconets that the bridge node belongs to. Considering questions (3) and (5), there may be a bottleneck problem in the scatternet if the node degree for the master or bridge nodes are high. There is one way to reduce the load on the master and bridge nodes. It is achieved by limiting the degree of the master and bridge nodes.

Consider questions (3) and (4). In the Bluetooth scatternet, different piconets are not synchronized to each other. Due to the loss of timing in the switching among different piconets, the bridge node may waste up to 2 time slots for each switch. Therefore, the degree of the bridge node and the number of the bridge nodes in one piconet should not be very large.

Question (2) raises a packet collision problem in the scatternet. Because all the piconets use the same 79 frequencies with different sequences, packet collisions may occur for the piconets that are within the same radio range. Packet collisions will degrade the network performance. Thus, we should control number of piconets if two masters from two different piconets are within the same radio range as they may transmit data packets using the same frequency.

In the Bluetooth specification [4], the slave can be in active, sniff, hold, or park mode. The mode of the slave in the scatternet is also an important issue in the Bluetooth scatternet formation algorithms. If all the slaves are always in active mode and most of them do not participate in the exchange of data, the network utilization will be reduced due to the polling of these active slaves.
The problem of node mobility is another important issue since Bluetooth is a wireless technology and most of the devices can move randomly. Mobility includes: node joining and leaving randomly. If the master leaves the network, how can the slaves maintain the connectivity with the network?

2.2 Related Work

Currently more and more research is focused on the scatternet formation algorithms in the literature. In this section, we introduce BTCP [5], Bluetrees [6], Bluenet [7], a scatternet formation algorithm proposed by Law and Siu [8], TSF [9], Bluestar [10], Loop Scatternet [11], various topologies proposed by Guerin et al. [12], Scatternet Route [13], and TPSF [14].

2.2.1 Bluetooth Topology Construction Protocol (BTCP)

BTCP is an asynchronous distributed protocol of Bluetooth scatternet formation [5]. The procedure of BTCP scatternet formation is shown in Figure 2.9 and is described as follows: (a) Initially all the devices collect the information of other nodes by alternative inquiry/inquiry scan. (b) The node with the highest Bluetooth device address is elected as the leader among all the nodes. The leader has the information of all nodes and determines a connectivity list, which includes the role of each node in the subsequent scatternet. (c) The leader connects the designated masters by paging them and transmitting the connectivity list. (d) The designated masters page their slaves according to the connectivity list. The bridge nodes wait to be paged by two masters. (e) The BTCP scatternet is formed.

BTCP causes a very small connection delay (about 2 seconds) if all the nodes start at the same instant. If the devices cannot start alternative inquiry/inquiry scan at the same time, the connection delay will become larger and the network cannot be guaranteed to be
Chapter 2 Overview and Related Work

BTCP assumes any two devices can reach each other so the coverage area of the scatternet is only one radio range. The resulting scatternet topology has the property that any two piconets are directly connected by a bridge node between them. The number of the nodes in the BTCP scatternet is limited to 36, corresponding to a maximum of 8 piconets.

2.2.2 Bluetrees

Bluetrees [6] can be employed to construct scatternets with low node mobility. The scatternet formation is initiated by a number of init nodes, which have the largest ID among all their neighbors. The init nodes form their own subtrees by paging their neighboring nodes. The scatternet is finally formed by combining the distributed subtrees together. By limiting the number of roles on each node and the number of slaves each master controls in one piconet, the switching cost is reduced in terms of switching delay,
scheduling, re-synchronization. The shortest path algorithm of network topology graph is used in the performance analysis of Bluetrees. The Average Route Error (ARE) is calculated by the difference between the actual path of Bluetrees and the shortest path. Simulation results show that the ARE is significantly influenced by the density of the network.

As shown in Figure 2.10, a Bluetrees scatternet has a hierarchical structure in which all the nodes except root nodes and leaf nodes are bridge nodes. Each bridge node acts as the slave of its upper layer piconets and master of its lower layer piconets. The routing is simple but it is not efficient since all the inter-piconet traffic has to go through the upper layer piconet and the upper layer can cause a bottleneck if the traffic load is high.

2.2.3 Bluenet

Bluenet [7] recognizes the disadvantages of the Bluetrees scatternet formation and proposed a “flatter” topology in which the bridge node can be either master/slave bridge or slave/slave bridge, as shown in Figure 2.11. The number of slaves in one piconet is limited to 5 in order to reduce the load on the master node. At the beginning, the master nodes are
selected randomly. Each master node invites \( n \) \( (n \leq 5) \) neighboring nodes to form one piconet by paging. In this way the network is formed by the separated piconets with some isolated nodes which are not connected by any master node. Then the isolated node initiates the combination of the distributed piconets by paging. Thus, the Bluenet scatternet is formed. The Average Shortest Path (ASP) and Maximum Traffic Flow (MTF) are used as the metrics of the performance evaluation. ASP is defined as the average path length between all the 2-node pairs. MTF is used to evaluate the traffic capacity which can be carried. The simulation results show that the ASP decreases by 0.23 when compared with Bluetrees algorithm and the MTF in Bluenet is higher than Bluetrees.

### 2.2.4 Scatternet Formation Algorithm Proposed by Law and Siu

In the scatternet formation algorithm of [8], all the nodes are assumed to be in one radio range. The isolated node, the piconet or the scatternet is called the network component. Each component has one leader. The scatternet is formed step by step by merging the network components. After each merge, only one leader remains in the merged scatternet and the other leader is retired. The formation algorithm is executed by each leader for merging the components. Figure 2.12 shows a scatternet formed by this algorithm.
Considering the scatternet is formed within one radio range, the number of piconets should be minimized. The formation algorithm makes the number of slaves as large as possible in each piconet so the resulting scatternet has the minimum number of piconets. Furthermore, the maximum degree of bridge nodes is limited to 2 considering inter-piconet switching costs. The resulting scatternet has only one leader. The leader in the scatternet can sense the incoming node by choosing inquiry or inquiry scan with a certain probability $p$ and the incoming node can also detect the master in the same way. This algorithm allows the new devices to join the scatternet dynamically. The devices that leave or fail are not considered in this algorithm.

### 2.2.5 Tree Scatternet Formation (TSF)

Similar to Bluetrees, TSF [9] also creates a scatternet with the tree structure. At any time, the topology created by TSF is a collection of one or more rooted spanning trees. At the
base of the traffic load, TSF assigns several slaves as the coordinators which are in *inquiry* or *inquiry scan* state to discover the neighboring trees. After two trees detect each other, the message will be sent from the coordinators to their root nodes. The merging of two trees is performed by one tree root node paging the other tree root node. Two neighboring trees can be merged only if the tree roots are in the radio range. TSF is self-healing in which nodes can join or leave at any time. However, the roots of the two neighboring trees cannot be guaranteed to reach each other. In that case, the scatternet cannot be guaranteed to be fully connected in a large coverage area.

### 2.2.6 Loop Scatternet Formation (LSF)

The LSF algorithm [10] assumes that all the nodes in the scatternet can reach each other. Much of the idea is borrowed from [8]. The scatternet is formed in two phases. The first phase is similar to [8]. A ring scatternet is formed by merging all the components. Each component may be an isolated device, a piconet or a ring scatternet. In the second phase, more bridges are created between some piconets in order to reduce the network diameter. Figure 2.14 shows an example of the Loop scatternet. To achieve a much shorter network
diameter, it needs more bridge nodes, which will increase the switching cost in the network. The performance analysis focused on the topology of the resulting network. Compared with [8], the number of piconets in the LSF scatternet is slightly larger but the network diameter is much smaller. It was shown in [10] that the scatternet will be fully connected even if the nodes do not power on at the same time.

2.2.7 Bluestar

Bluestar [11] creates the scatternet in three phases, namely neighbor discovery, neighbor grouping, and role assignment. In the neighbor discovery phase, it applies the symmetric method in which the nodes perform the alternative inquiry and inquiry scan. The asymmetric method will also be used if a node wants to join the network. The asymmetric method works as follows: the nodes in the network perform periodic inquiry scan and the incoming node performs inquiry. It assumes that the isolated node knows if it is in the network setup period or if it is joining the network so that either the symmetric or asymmetric method may be selected. In the neighbor grouping phase, the neighboring nodes are grouped into a number of sets and a link is established to one of the nodes in each set. The two nodes belong to the same set if there exists a non-neighbor
intermediate node connecting them. In the third phase, the role of the device (either as a master or as a slave) is determined based on the selected neighbors to which a connection is to be established. Figure 2.15 shows an example of Bluestar scatternet.

The simulation results in [11] indicate that Bluestar outperforms the tree topology scatternet in terms of communication distance, reachability, and number of links.

2.2.8 Various Topologies Proposed by Guerin et al.

In [12], various scatternet formation algorithms are proposed. First of all, it shows that DSF (Depth First Search) algorithm can provide the minimum node degree when compared with MST (Minimum weighted Spanning Tree) and BFS (Breadth First Search). The author also proposed an EMDST (Extended Minimum Degree Spanning Tree) algorithm, which provides the means for tuning the degree constraints on the master and bridge nodes. In addition, a fully distributed and dynamic algorithm is proposed and it is shown that the algorithm works well in terms of the number of edges, the average degree of the nodes, and the number of nodes with a dual role.

2.2.9 Scatternet Route

An on-demand scatternet route formation algorithm was proposed in [13], as shown in
Figure 2.16. A scatternet is only created when communication is required. The flooding method is used to search a route from the source to the destination. The on-demand scatternet is formed via paging from the destination node to the source node. The route discovery message has to be carried by two inquiry ID packets because the BD_ADDR (48 bits) is used to uniquely identify the source, intermediate and destination nodes.

In flooding, each node has to take part in the route discovery by inquiry even if the source node and destination node are neighbors. The flooding method may cause a high number of unnecessary packet rebroadcasts in the network. Therefore, the flooding that occurs in finding the route is not efficient and wastes precious limited bandwidth.

In particular, without any knowledge of the neighborhood nodes, Bluetooth devices use the inquiry procedure to flood the routing queries once it receives a new inquiry message from another node. Each inquiry message has to be kept for at least 2.56 s in order to guarantee that the neighboring nodes can receive it. The network may become saturated because each node has to take part in the route discovery for each communication session.
2.2.10 Two-Phase Scatternet Formation (TPSF) Algorithm

To avoid the bottleneck problem in the master and bridge nodes, TPSF [14] was designed such that signalling is separated from its associated traffic path by placing the signalling on the control scatternet and the traffic on the on-demand scatternet.

The procedure for the scatternet formation is shown in Figure 2.17. The node with the highest number of neighbors among all its neighbors is selected as the master node in the control scatternet and the master node pages all its neighbors one by one to set up the scatternet. The bridge node is the first node paged by two masters. The bridges are in active mode all the time and all the pure slave nodes are in park mode in the control scatternet. After the control scatternet has been created, the on-demand scatternet formation is initiated by the source node that wants to transmit data to the destination node. It has two steps. First, the Dynamic Source Routing (DSR) [15] based scheme is applied for the piconet route discovery in the control scatternet. Then all the master nodes along the piconet route select the participating nodes for the on-demand scatternet. The participating nodes form the path information, which is sent again from the master of the source node to the destination node. The destination node can initiate the connection setup by the paging procedure. In this scheme, the master node uses the information from its own slaves and the information from the adjacent piconets to choose the participating nodes. This requires that each slave periodically updates the neighbor information and each master updates adjacent piconets' neighbor information. If there is no application using the node, updating the neighbor information periodically may be wasteful especially in Bluetooth where power and bandwidth are limited. The updating period depends on the mobility of the nodes in the network.

In the formation of an on-demand scatternet, the control scatternet is used to find
Chapter 2 Overview and Related Work

(a) Distributed nodes without connections
(b) The control scatternet is constructed
(c) RREQ/RREP/PREQ/PREP are exchanged
(d) On-demand scatternet is constructed

Figure 2.17 A Two Phase Scatternet Formation Protocol [14]

the shortest route to create an on-demand scatternet. The on-demand scatternet is dedicated to the traffic transmission so that high throughput can be achieved. By simulation, TPSF has been shown to achieve a much higher throughput than BTCP.

2.3 Comparisons of Previous Scatternet Formation Algorithms

All the previously mentioned scatternet formation algorithms can be classified into two categories. The first one is called the proactive scatternet formation, which means the scatternet is built as soon as the devices are powered on ([5] - [12]). The second one is called the reactive scatternet formation algorithm, which means the scatternet is formed on demand of traffic requirement [13][14]. A comparison of different proactive scatternet
Table 2.2 A Comparison of Proactive Scatternet Formation Algorithms

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Tree Topology</th>
<th>Coverage Area that is within one radio range</th>
<th>Support of Mobility</th>
<th>Formation Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTCP [5]</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Centralized</td>
</tr>
<tr>
<td>Bluetrees [6]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Distributed</td>
</tr>
<tr>
<td>Bluenet [7]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Distributed</td>
</tr>
<tr>
<td>Law and Siu [8]</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Distributed</td>
</tr>
<tr>
<td>TSF [9]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Distributed</td>
</tr>
<tr>
<td>Bluestar [10]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Distributed</td>
</tr>
<tr>
<td>LSF [11]</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Distributed</td>
</tr>
<tr>
<td>Guerin et al. [12]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Distributed</td>
</tr>
</tbody>
</table>

formation schemes is summarized in the Table 2.2. Among them, three algorithms [5][8][11] assume that all the nodes are in the same radio range. In that case, the number of piconets and network diameter should be made as small as possible. Moreover, if two nodes that are within the same radio range need to communicate, it may be better to set up a new piconet that contains only these two nodes. If the coverage area of the scatternet is within one radio range, the more the number of piconets, the higher the chance of frequency collisions. Therefore, minimizing the number of piconets is an important issue for the scatternet created within the same radio range.

Compared with the reactive scatternet formation algorithms, the proactive scatternet formation algorithm [5] - [12] require the periodic maintenance which definitely consumes the limited bandwidth. The maintenance can be wasteful sometimes if there is no traffic requirement. In addition, most of the proactive scatternet formation algorithms do not consider the support of mobility. In BTCP, all the nodes are required to power on at
the same time, otherwise the network connectivity cannot be achieved.

Compared with the proactive scatternet formation algorithms, the reactive scatternet formation algorithms [13][14] are more adaptive to the dynamic environment and they can achieve higher throughput because the scatternet is dedicated to the traffic requirement.

2.4 Problem Definition

After reviewing several scatternet formation algorithms proposed in the literature, several important questions are being raised:

(1) How to maintain and update the network topology information in a dynamic environment?

(2) How to solve the bottleneck problem that exists at the master and bridge nodes?

(3) How to determine an optimal route from the source node to the destination node for a single communication session?

Considering question (1), the proactive scatternet formation algorithms require the nodes to periodically keep track of the network topology. However, in Bluetooth PAN where the mobile nodes have both power and bandwidth constraints, conventional proactive scatternet formation algorithms may not be suitable.

In the Bluetooth specification [4], each master node can have at most 7 active slaves and the maximum number of the piconets that the bridge node can belong to is not addressed. To provide an answer for question (2), most of the scatternet formation algorithms focused on the degree of the master and bridge nodes because the master and bridge nodes can easily become the bottlenecks in the scatternet. These bottlenecks will
affect the throughput and the end-to-end delay. In some scatternet formation algorithms (i.e. [5][8][11][14]), the degree of the bridge node is limited to 2. In Bluetree [6] and Bluenet [7], the degree of the master node is limited to 5. The two-phase scatternet formation algorithm (TPSF) [14] has been proposed to reduce the high traffic load at the master and bridge nodes by separating the control traffic and data traffic into a control scatternet and an on-demand scatternet respectively. An on-demand scatternet is only formed when the source node wants to exchange data with the destination node. TPSF can create a scatternet with high aggregate throughput and low end-to-end delay.

To support the communication between a source node and a destination node, the route discovery problem is an important issue in Bluetooth scatternet formation algorithms, as mentioned in question (3). Some scatternet formation algorithms (i.e. [6][9]) create a tree topology in order to simplify the route discovery. However, bridge nodes may cause a bottleneck problem in the tree topology. In the proactive scatternet formation algorithms, the route maintenance is required for the dynamic environment. To overcome the redundant work in maintaining route discovery, a reactive protocol has been designed in [13], which uses flooding to propagate the routing discovery messages. In flooding, each node receiving the route discovery message has to propagate to its neighbors. In TPSF, the scatternet for the data communication is called an on-demand scatternet. To select an on-demand scatternet route, the masters in the control scatternet use the route information collected when the slaves access the network. It is assumed that the slaves do not move after connecting to the control scatternet in TPSF.

An important aspect of Bluetooth scatternet formation algorithm is to support the dynamic operation [14]. In TPSF, the problem is that in a dynamic environment in which the node may change its position with time, the route information collected when the
nodes access the network may not be up-to-date. Although the route information is discovered on-demand in the scatternet route, the flooding increases the load in each node and the possibility of network congestion because of the redundant route discovery traffic.

The scatternet formation algorithm proposed in this thesis is called TPSF+. The goal of TPSF+ is to support node mobility within the scatternet. The on-demand route information is obtained when the source node wants to communicate with the destination node. This is contrary to TPSF, where the on-demand scatternet route information is obtained when the nodes initially access the network. TPSF+ allows the slaves to move randomly around the master in the control scatternet without the additional control update information. For performance comparison, we conducted simulation experiments on TPSF, TPSF+, and BTCP.
Chapter 3  A New Two-Phase Scatternet Formation (TPSF+) Algorithm

In this chapter, we extend TPSF to TPSF+. In our proposed TPSF+, the first phase (i.e. the control scatternet formation), is the same in TPSF. We proposed an alternative scheme called proactive-reactive scatternet formation algorithm for the second phase of TPSF. In TPSF+, the control scatternet is utilized more efficiently for route discovery.

The rest of this chapter is organized as follows. The control scatternet formation in the previous TPSF is introduced in Section 3.1. Our proactive-reactive on-demand scatternet formation algorithm is proposed in Section 3.2. The formats of the newly defined packet types are introduced in Section 3.3. A summary is given in Section 3.4.

3.1 Control Scatternet Formation

In TPSF+, the control scatternet formation is the same as in TPSF. Once powered on, each node discovers its neighbors and then exchanges the neighboring information by alternating inquiry and inquiry scan continuously. Then the node with the largest number of neighboring nodes among all its neighboring nodes is selected as the master node. The master node pages all of its neighbors one by one to set up the control scatternet. There are at most one bridge node between any two masters. In the control scatternet, the bridge nodes are in active mode all the time. Each bridge node belongs to two piconets and acts as the slave node in both of its piconet. The pure slave nodes are in park mode in the control scatternet. After the control scatternet is formed, the masters in the neighboring piconets exchange the neighboring information. In TPSF, the neighboring information contains the BD_ADDRs of the slaves in the neighboring piconets. Whereas in TPSF+, this neighboring information contains BD_ADDRs and the corresponding PM_ADDRs of the slaves in the neighboring piconets.
3.2 On-demand Scatternet Formation

Our TPSF+ focuses on designing an alternative scheme for the second phase in TPSF. After the control scatternet is formed, each master in the control scatternet would have collected the mapping between BD_ADDR and PM_ADDR for the slaves in its own piconet and its neighboring piconets.

The second phase of TPSF+ has two procedures. The first procedure is for masters’ route discovery. In this procedure, the masters’ route is chosen based on source routing when the source node wants to communicate with a destination node. The second procedure is for on-demand scatternet route discovery and connection set up. In this procedure, the on-demand scatternet route is defined as a list of nodes in the final on-demand scatternet. The inquiry procedure is used to find an on-demand scatternet route, which is initiated by the source node. The search is restricted to those piconets along the masters’ route. After the on-demand scatternet route is discovered, the on-demand scatternet can be set up from the destination node to the source node by the paging procedure.

3.2.1 Masters’ Route Discovery Procedure

A packet that is sent over the Bluetooth scatternet may need to go through one or several piconets to get from the source to the destination. The master’s route is defined as a list of the piconets along the communication path and is identified by a list of masters’ BD_ADDRs. The masters’ distance is defined as the number of piconets that the packet traverses from the piconet of the source node to the piconet of the destination node. The masters’ route discovery is performed among the master nodes and bridge nodes in the control scatternet. The steps of the masters’ route discovery procedure are as follows:

(i) When a source node needs to communicate with another node in the network, it sends a masters’ route request packet (MRREQ) to its master. The MRREQ sent by the source node contains the BD_ADDR of the source node and the destination node, and the
communication session number (CSN) generated by the source node. The BD_ADDR of the source node and the CSN can uniquely identify a communication session.

(ii) When the master node receives the MRREQ packet, it first updates the packet. Then it sends the MRREQ packet to all of its bridge nodes. The bridge nodes simply forward the MRREQ packet to their other masters. The MRREQ packet contains the list of master nodes that the MRREQ packet has traversed, the BD_ADDR/PM_ADDR mapping information of the slaves which move in or out of the piconet, and the maximum masters’ distance, which is the maximum number of masters’ distance that the MRREQ packet can traverse. Whenever the master receives the MRREQ packet, it will update the MRREQ packet by inserting its address and BD_ADDR/PM_ADDR mapping information of its incoming slaves and outgoing slaves, and by decrementing the maximum number of masters’ distance by 1.

(iii) When the MRREQ packet reaches the master of the destination node, the masters’ route response (MRREP) packet, which contains the masters’ route information, is sent back on the reverse path. On the reverse path, when the MRREP passes through the master node along the masters’ route, the master node will update its routing table. The master node along the route will also broadcast the route information to its slaves. This route information contains the masters’ route, the mapping information of BD_ADDR/PM_ADDR for the incoming and outgoing slaves in its piconet and its upstream piconets. After receiving the route information, each slave node can decide by itself whether it will be the candidate node or not, depending on the number of active communication sessions it belongs to. The candidate node is defined as the node which can participate in the on-demand scatternet route discovery. For example, if the slave is idle, then it will become the candidate node. For those slaves which belong to another on-demand scatternet will simply accept the route information but will not become the candidate nodes. Therefore,
after receiving the route information from the master, each candidate node will know the masters’ route for a particular communication session and the PM_ADDR/BD_ADDR mapping information in its piconet and in its upstream piconet along the route. For a particular communication session, the BD_ADDR/PM_ADDR mapping information will be used by candidates to obtain the BD_ADDR from PM_ADDR in the on-demand scatternet route discovery.

In Figure 3.1, M1, M2 and M3 denote master node 1, master node 2, and master node 3 respectively. S and D denote the source node and the destination node respectively. N denotes CSN generated by S. MRREQ0, MRREQ1, and MRREQ2 are presented as the MRREQ packet created or updated by S, M1, and M2 respectively. When S needs to communicate with D, S sends MRREQ0 to M1. After receiving the MRREQ0, M1 updates MRREQ0 to MRREQ1 and sends MRREQ1 to all of its bridge nodes. When bridge node B1 receives the MRREQ1, it will then forward the MRREQ1 to M2. M2 notices that node D is not in its piconet, so it updates MRREQ1 to MRREQ2 and sends MRREQ2 to bridge node B2. Bridge node B2 simply forwards MRREQ2 to M3. M3 will not update MRREQ2 because M3 knows that D is its slave. On the reverse path, M3 sends back to B2 a masters’ route response (MRREP) packet, which contains the masters’ route (M1, M2, M3). After each master along the route receives the MRREP packet, the routing table will be set up for this particular communication session, which is identified by the BD_ADDR of the source node and CSN. The mapping information of BD_ADDR/PM_ADDR for the slaves in the upstream piconet and in its own piconet has been stored in each master node along the route. M3, M2 and M1 can broadcast the route information to all of their slave nodes. The route information contains the masters’ route information and the mapping information. Table 3.1 shows the masters’ routing table saved in M1, M2, and M3 for a particular communication session identified by (S, N). Tables 3.2 - 3.4 show
Figure 3.1 Masters’ Route Discovery Procedure

MRREQ0: (S, D, N)
MRREQ1: (S, D, N, M1, Updated BD_ADDR/PM_ADDR)
MRREQ2: (S, D, N, M1, M2, Updated BD_ADDR/PM_ADDR)
MRREP: (S, D, N, M1, M2, M3)

Masters’ Route: (M1, M2, M3)
Table 3.1 Masters’ Routing Table

<table>
<thead>
<tr>
<th>Source</th>
<th>CSN</th>
<th>Masters’ Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>S</td>
<td>N</td>
<td>M1, M2, M3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 3.2 Mapping Table in Piconet M1

<table>
<thead>
<tr>
<th>BD_ADDR (48 bits)</th>
<th>PM_ADDR (8 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>S</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 3.3 Mapping Table in Piconet M2

<table>
<thead>
<tr>
<th>BD_ADDR (48 bits)</th>
<th>PM_ADDR (8 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 3.4 Mapping Table in Piconet M3

<table>
<thead>
<tr>
<th>BD_ADDR (48 bits)</th>
<th>PM_ADDR (8 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

the mapping information for BD_ADDR/PM_ADDR of piconet M1, M2, and M3 respectively. M3 broadcasts the route information from Tables 3.1, 3.3 - 3.4 to its slaves. M2 broadcasts the route information from Tables 3.1 - 3.3 to its slaves. M1 broadcasts the information from Tables 3.1 - 3.2 to its slaves.
3.2.2 On-demand Scatternet Route Discovery and Connection

In this procedure, after the masters' route is discovered, the on-demand scatternet route is decided among the slaves along the masters' route. Thus, unnecessary route discoveries can be avoided. The on-demand scatternet route discovery procedure is used to determine which particular nodes should be chosen to form the on-demand scatternet. Only the candidate node will process the on-demand scatternet route request (OSRREQ) packet and flood the OSRREQ packets to other candidate nodes by inquiry. After the on-demand scatternet route is discovered, the on-demand scatternet connection can be set up by the paging procedure.

3.2.2.1 PM_ADDR Scheme

The OSRREQ packet carried by the inquiry ID packet can only be up to only half a timeslot long as defined in the Bluetooth specification [4]. As suggested in [13], the general route request packet contains the following fields:

< access code (72 bits), source BD_ADDR (48 bits), destination BD_ADDR (48 bits), upstream node BD_ADDR (48 bits), the hop limit (> 4 bits), packet session number (>4 bits) >

Using the fields above, one inquiry packet is not long enough to transmit one OSRREQ packet. We choose the masters' distance (8 bits) and PM_ADDR (8 bits) to identify the upstream node instead of the upstream node BD_ADDR (48 bits). With this modification, one inquiry ID packet can be used to transmit an OSRREQ packet. Our proposed OSRREQ packet contains the following fields:

< access code (72 bits), source BD_ADDR (48 bits), masters' distance of upstream node (8 bits), upstream node PM_ADDR (8 bits), hop limit (8 bits), intra-piconet hops (8
bits), CSN (8 bits) >

In our proposed OSRREQ packet, the pair < source BD_ADDR, CSN > generated by the source node is used to identify one communication session. The pair < masters' distance, upstream node PM_ADDR > in OSRREQ packet can be used to identify an upstream node in a particular communication session. Once a candidate node receives an OSRREQ packet from its upstream node, it knows which piconet the packet comes from by checking the masters’ distance in the received OSRREQ. If the received OSRREQ has the same masters’ distance number with the candidate node, the OSRREQ packet comes from another candidate node in the same piconet. If the masters’ distance number of the received OSRREQ packet is smaller than the candidate node, then the OSRREQ packet is from an upstream piconet along the masters’ route. Otherwise, the candidate node will simply drop the OSRREQ packet. By checking the mapping information of BD_ADDR/PM_ADDR, the receiving candidate node can determine which upstream node sent this OSRREQ packet. In this way, (masters’ distance number, PM_ADDR) can be used to identify the upstream node instead of using the BD_ADDR 48 bits. Thus, 32 bits are saved.

3.2.2.2 Maximum Intra-piconet Distance

The intra-piconet distance is defined as the number of hops that the OSRREQ packet has traversed within the same piconet in a control scatternet. The maximum intra-piconet distance is equal to $K$ in order to limit the number of hops in the resulting on-demand scatternet. Each time the OSRREQ message reaches the candidate node in the downstream piconet along the masters’ route, the maximum intra-piconet distance is set as $K$. Each time the OSRREQ message reaches a candidate node in the same piconet, the number of intra-piconet hops is decreased by 1.
3.2.2.3 On-demand Scatternet Formation Procedure

The steps of the on-demand scatternet formation procedure are as follows:

(i) On-demand scatternet route discovery and reverse routing table establishment:
The source node S sends the OSRREQ packet by inquiry. After receiving the OSRREQ packet, the candidate node will first check the source BD_ADDR, the CSN, and the intra-piconet distance in the OSRREQ packet. If the candidate node receives the same source BD_ADDR and CSN as the previous OSRREQ, or if the number of intra-piconet hops is 0, it will simply discard the OSRREQ packet. The candidate node first updates its reverse routing table by recording the source BD_ADDR, CSN and next hop information for the source node. This reverse routing entry may later be used for establishing the connection back to the source node. The next hop information for the source node is the BD_ADDR derived from the upstream node PM_ADDR in the OSRREQ packet by using the BD_ADDR/PM_ADDR mapping tables. The candidate node also updates the OSRREQ packet by modifying the masters' distance to its own masters' distance, modifying the PM_ADDR to its own PM_ADDR, decreasing the hop limit by 1, and decreasing the intra-piconet distance by 1 or setting the intra-piconet distance to be *K*. Finally, the updated OSRREQ packet is broadcast by inquiry.

(ii) Reverse connection setup: After the destination node D receives the OSRREQ, it simply pages its upstream candidate node. The upstream candidate node will then page its upstream candidate node using the reverse route entry until the source node is being paged. Thus, the connection can be set up from the destination node to the source node. The on-demand scatternet is now created. The candidate nodes which join the on-demand scatternet are called the participants of the on-demand scatternet. In the resulting on-demand scatternet, all the intermediate participants are the bridge nodes, which act as the master of its upstream node and as the slave of its downstream node.
For example, in Figure 3.2, the numbered slave nodes are the candidate nodes taking part in the on-demand scatternet route discovery, assuming $K$ is equal to 3. The OSRREQ $(S, N, 1, 1, 3, 1)$ sent by candidate node 3 is received by candidate node 4. Candidate node 4 has the information of Tables 3.1 - 3.3 obtained from M2 during the masters’ route discovery procedure. By checking the masters’ distance in $(S, N, 1, 1, 3, 1)$, node 4 will know $(S, N, 1, 1, 3, 1)$ is sent from the upstream piconet M1 because the masters’ distance 1 in $(S, N, 1, 1, 3, 1)$ is the masters’ distance of node 4’s upstream piconet. By checking the mapping information in Table 3.2, candidate node 4 will know that the BD_ADDR of the sender is 3 from the PM_ADDR 1 in the $(S, N, 1, 1, 3, 1)$. In this way node 4 will easily know the BD_ADDR of its upstream node is 3. In the same way, node D knows its upstream node is node 6. Node 6 knows its upstream is node 5. Node 5 knows its upstream is node 4. Node 3 knows its upstream is node 2. Node 2 knows its upstream is node S. Finally the connection can be set up by D paging upstream node 6, and node 6 pages its upstream 5 until node 2 pages the source node S.

In the worst case, the on-demand scatternet route may not be found among the slaves in the control scatternet because the OSRREQ packet from the candidate nodes in one piconet cannot reach any candidate node in its downstream piconet. In that case, the on-demand scatternet route can simply go through the master nodes and bridge nodes along the masters’ route in the control scatternet. As shown in Figure 3.3, in this case the on-demand scatternet route would be $(S, M1, B1, M2, B2, M3, D)$.

### 3.3 Packet Formats

In this section, the newly defined packet types for the masters’ route discovery and on-demand scatternet route discovery are described.
On-demand Scatternet Route Request (OSRREQ) Packet:

\[(S, N, x_1, x_2, x_3, x_4)\]

- Source BD_ADDR
- Intra-piconet distance
- Masters' distance
- Communication Session Number
- PM_ADDR of the upstream node

Figure 3.2 On-demand Scatternet Route Discovery Procedure
As we have mentioned above, there are two control packets - MRREQ and MRREP in the masters' route discovery procedure. In the baseband layer of Bluetooth, we define the masters' route request (MRREQ) packet to be a DM packet because the payload of the packet is the route information and should be protected by FEC. In the Bluetooth baseband layer, there are DM1, DM3 and DM5 packets; Therefore the decision of which particular packet type is to be used depends on the length of the MRREQ and MRREP packets.

### 3.3.1 MRREQ Packet Format

In addition to discovering and collecting the masters' route information, the MRREQ is also used to transmit the BD_ADDR/PM_ADDR mapping information to the downstream piconet. The format of the MRREQ packet is shown in Figure 3.4.

- **VER (3 bits):** Version number.
### Figure 3.4 MRREQ Packet Format

- **Flag (5 bits):** If the first bit is set to 0, then it is the control packet. Otherwise, it is the data packet [15].

- **Masters’ Distance (8 bits):** The Masters’ Distance field carries the maximum number of masters’ distance that the route request packet can traverse. It is decreased by 1 as the MRREQ packet traverses a piconet. Once the Masters’ Distance field reaches the value of 0, the node should discard the packet.

- **Packet Type (8 bits):** This field is equal to 1 for MRREQ packet.
• CSN (8 bits): The communication session number is generated by the source node to identify one particular communication session. If the CSN has been seen before, the receiving node simply discards the received MRREQ.

• Incoming Number (8 bits): Number of the slaves that has joined this particular piconet after the last MRREQ was sent.

• Outgoing Number (8 bits): Number of the slaves has left this particular piconet after the last MRREQ was sent.

• Source BD_ADDR (48 bits): Bluetooth device address of the source node.

• Destination BD_ADDR (48 bits): Bluetooth device address of the destination node.

• Master 2 (48 bits): The 2nd master’s BD_ADDR along the masters’ route.

• Master n - 1 (48 bits): The (n-1)th master’s BD_ADDR along the masters’ route.

• BD_ADDR 1 (48 bits): The BD_ADDR of the first incoming slave.

• BD_ADDR m (48 bits): The BD_ADDR of the mth incoming slave.

• PM_ADDR 1 (48 bits): The PM_ADDR of the first incoming slave.

• PM_ADDR m (48 bits): The PM_ADDR of the mth incoming slave.

• PM_ADDR m+1 (48 bits): The PM_ADDR of the first outgoing slave.

• PM_ADDR m+k (48 bits): The PM_ADDR of the kth outgoing slave.

The header of the MRREQ packet includes VER, flag, masters’ distance, packet type, CSN, incoming number, and outgoing number. The packet length is determined by masters’ distance (= n), incoming number (= m), and outgoing number (= k). The total length of the packet is \(48 \times (1 + n + m) + 8 \times (m + k)\) bits long.
3.3.1.2 MRREP Packet Format

MRREP is the response packet of MRREQ and provides the masters' route information for a particular communication session. The format of MRREP packet is shown in Figure 3.5.

- Packet Type (8 bits): This field is equal to 2 if it is an MRREP packet.

The semantics of the other fields shown in Figure 3.5 are the same as in the MRREQ packet. The total length of the MRREP packet is $32 + 48 \times n$ bits long.

3.3.2 On-demand Route Request (OSRREQ) Packet

In the Bluetooth baseband layer, the OSRREQ packet is carried by an inquiry ID packet. By using the PM_ADDR we proposed, the length of the packet can be reduced to 130 (= $72 + 24 + 8 \times 4 + 2$) bits. Therefore, only one inquiry ID packet is used to transmit the OSRREQ packet. The OSRREQ packet format is shown in Figure 3.6.

- Access Code (72 bits): It is defined in the Bluetooth specification [4].
Chapter 3 A New Two-Phase Scatternet Formation (TPSF+) Algorithm

Figure 3.6 OSRREQ Packet Format

- Source LAP (24 bits): The lower address part of the source BD_ADDR.
- CSN (8 bits): The communication session number generated by source node is used to identify a particular communication session.
- Masters’ Distance (8 bits): This field is the number of piconets from the source node to the current piconet along the masters’ route.
- PM_ADDR (8 bits): The parked mode member address of the node sending the packet as defined in the Bluetooth specification [4].
- Hop Limit (8 bits): The number of hops from the source node to the node which generates the packet.
- Intra-piconet Distance (2 bits): It carries the maximum number of hops the route request packet can traverse in one piconet. It is set as $K$ when the OSRREQ packet reaches the node of a new piconet. It is decreased by 1 as the OSRREQ packet traverses the nodes in the same piconet. Once it reaches the value of 0, the node should discard the packet.

The source LAP field and CSN can uniquely identify a particular communication session. The masters’ distance field can identify a unique piconet of the control scatternet for a particular communication session. The PM_ADDR can be used to identify a particular slave in a particular piconet.
3.4 Summary

In this chapter, we have proposed a new algorithm called TPSF+ for creating the on-demand scatternet in TPSF. It is clear that the masters’ route leads to a “reduced” set of slaves participating in the on-demand scatternet route discovery. Furthermore, instead of all the slaves along the masters’ route taking part in the on-demand scatternet route discovery, the on-demand scatternet route discovery is performed by selective flooding, depending on whether there exists a communication session in the slaves. To reduce the number of hops in the on-demand scatternet, the on-demand scatternet route discovery traverses in one piconet with at most $K$ hops, where $K$ is the maximum intra-piconet distance. In addition, we use the PM_ADDR to identify a node instead of using BD_ADDR in order to reduce the length of on-demand scatternet route request (OSRREQ) packet. As result, a single inquiry ID packet can accommodate all the required information for the on-demand scatternet route discovery. Finally, in the case when the source node cannot reach the destination node through the slaves along the masters’ route, we propose to use the master nodes and bridge nodes as the participating nodes in the on-demand scatternet route.

Because the participating nodes are discovered on demand, the slaves can randomly move around their masters in the control scatternet without any additional control message. Our scheme can achieve a higher network utilization and is more suitable in a dynamic environment.
Chapter 4 Simulation Model and Performance Analysis

In order to investigate the performance of our proposed TPSF+, the bluescat simulation model based on ns-2 is used and several extension models are designed. In Section 4.1, an overview of ns-2 and Bluescat simulator is given. In Section 4.2, the extension models are described. In Section 4.3, we present the simulation results for performance analysis.

4.1 Bluescat Simulation Model on ns-2

Bluescat simulator [16] is an ns-based simulator for Bluetooth protocol. As an open source Bluetooth network simulator, Bluescat is the extension of BlueHoc network simulator [17], which is developed based on ns-2 [18] by IBM. Bluescat has followed but not fully implemented part of the Bluetooth specification 1.1 [4].

In Bluescat, the Bluetooth node is derived from the node in ns-2 by including the baseband layer and link layer in the Bluetooth specification [4]. Figure 4.1 shows the structure of a Bluetooth node in Bluescat. Bluetooth baseband and link layers are composed of several sub-objects, from lower layer to higher layer, called baseband, MAC Scheduler, ARQ module, LMP, Leaky Bucket Filters, L2CAP, and Bluetooth Host. The baseband provides basic features of Bluetooth baseband. In MAC layer, Sched_ is used to implement the Deficit Round Robin (DRR) scheduling model [19]. LMP implements some LMP signalling commands to establish an Asynchronous Connection-Less (ACL) link between the master and the slave once paging is over and the slave is synchronized to the master’s hopping sequence. LMP also passes the QoS parameters to the scheduler, which determines whether the connection can be accepted. If the QoS negotiation at the LMP is successful, the L2CAP layer establishment can be performed. L2CAP performs segmentation and reassembly (SAR) for higher layer packets. BTHost is layered over L2CAP and provides an interface between the Bluetooth node data link layer and the
transport layer. The unnumbered fast ARQ module is implemented on the link control layer in Bluescat. All the transport layer and application layer modules are implemented in the original ns-2 model. The application names such as FTP and Telnet have been set up in BTHost in order to make the node recognize the applications.

When an incoming packet arrives, it passes from baseband to BTHost and then arrives at the node entry, the addr_mux is used to determine the next node to which the
packet should be sent. The port_dmux is used to distribute the packet to the correct agent - transport layer. The Bluetooth node in Bluescat is an aggregate object used to implement the function of a Bluetooth device according to the Bluetooth specification [4].

4.2 Extension to Bluescat
Bluescat is capable of simulating the Bluetooth scatternet connection. However, the data communication on the scatternet, the support of mobility, and packet collisions have not been implemented. To make the simulation results more practical, we extend the Bluescat model by including data communications function on the bridge node, the packet collisions model, and the mobility model. The purpose of these extensions is to show the basic features of our proposed TPSF+ algorithm.

4.2.1 Data Communications on Bridge Nodes
In general, in the Bluetooth specification [4], each L2CAP channel connects one communication session between one master node and one slave node. The L2CAP channel is referred by a channel identifier (CID). In Bluescat simulator, with only one CID the bridge node can only communicate with its first connected master. To make the bridge node communicate with all of its masters, we used multiple CIDs on the bridge nodes, as shown in Figure 4.2. As addressed in the specification [1], each of the CIDs is a local name representing a logical channel to each master. In this way, the bridge node can communicate with all of its masters. For example, in our simulation, each bridge has only two masters; it uses two CIDs for the data transmission.

4.2.2 Packet Collisions Model
Packet collision occurs when a node receives two packets carried by the same frequency channel. Because different piconets use the same 79 frequencies with different sequences,
the higher the number of piconets in the same radio range, the higher the chance of the frequency collisions. Although Bluetooth uses a technique called spread-spectrum frequency hopping and packet collisions can be resolved by retransmission, the performance can still be affected by packet collisions. Therefore, it is necessary to build the interference model in order to show how much the performance is affected by packet collisions. Previous error model in Bluescat is based on estimation of error probability, depending on the distance between the two nodes. We modified the error model based on frequency interference in the baseband layer. Our interference model is shown in Figure 4.3. In our model, each node has two queues called the potential interference queue (potinterferQ_) and the interference queue (interferQ_). During the receiving period of one node, the potinterferQ_ is used to store all the currently used frequencies by other nodes in the radio range. At the beginning of the receiving period, if the frequency on which the receiver is listening to is found in the potinterferQ_, the frequency will be stored in the interferQ_. During the receiving period, if another packet is transmitted on the same frequency on which the receiver is listening to, the frequency will also be stored in the interferQ_. At the end of the transmission period, the receiver can determine whether the
Receiver: $R_1, f_x$

Figure 4.3 Interference Model

collision has occurred by checking if the interferQ_ is not empty or not. If the interferQ_ is empty, then the packet has correctly been received. Otherwise, frequency collision has occurred and then the packet should be retransmitted. The interferQ_ is used to store the frequencies which will cause collisions. The length of the interferQ_ shows the intensity...
4.2.3 Mobility Model

We also created a node-movement generator on the baseband of Bluetooth. The model we used is called the random waypoint mobility model [20]. In this model, the Bluetooth node moves from its current location to a new location by randomly choosing a direction and speed in which to travel. The new speed and destination are chosen from the pre-defined ranges, [speedmin, speedmax] and [(minxpos, minypos), (maxxpos, maxypos)] respectively. Each movement in the mobility model occurs after a constant time interval called pausetime, at the end of which a new direction and speed are chosen.
4.3 Examples of the Scatternet Topology

Figure 4.4 shows an example of a control scatternet which consists of a single piconet. Figure 4.5 shows an example of a control scatternet which consists of two piconets. The radius of each piconet in the control scatternet is 10 meters. The position of each slave is placed randomly in the coverage area of each piconet. The source and destination nodes are also selected randomly. In Figure 4.4, an on-demand scatternet with one-hop connection was created because the source and destination nodes are within the radio range. In Figure 4.5, an on-demand scatternet with two-hop connection was created by TPSF+.

4.4 Performance Analysis

We use several performance metrics for the performance evaluations of TPSF+. In Section 4.4.1, we analyze the throughput and end-to-end delay with packet collisions by compar-
ing TPSF+ and BTCP. In Section 4.4.2, we determine the intra-piconet distance, the average hop distance, the connection delay, and the path successful ratio. The performance comparison between TPSF and TPSF+ is given in Section 4.4.3.

4.4.1 Throughput and End-to-end Delay with Packet Collisions

By simulation, we determine the aggregate throughput and end-to-end delay as a function of the number of communication sessions. In our simulation model, 36 nodes are randomly placed in an area within the same radio range. Then for each communication session, we randomly select 2 nodes as the source node and the destination node. For TPSF+, we assume that each on-demand scatternet is for one communication session. We compare our TPSF+ scheme with BTCP with the occurrence of packet collisions. For each data point, the simulation was run 10 times and each run time was 200 s.

For UDP performance, we used two different packet types - DH3 and DH5. In the simulation, we applied the CBR (Constant Bit Rate) traffic with maximum bit rates which are 585.6 kbps for DH3 packets and 723.2 kbps for DH5 packets. Figures 4.6 and 4.8 show the aggregate throughput with respect to the number of the communication sessions using TPSF+ and BTCP. Figures 4.7 and 4.9 show the average end-to-end delay comparison between TPSF+ and BTCP, with the packet type of DH3 and DH5, respectively. The end-to-end delay for UDP performance is determined from the time when the packet is created to the time when the packet is received on the application layer.

For both DH3 and DH5 packets, the aggregate throughput increases when the number of communication session increases. The aggregate throughput is much higher in TPSF+ than in BTCP. The reason is that different scatternets are created for different communication sessions in TPSF+ and BTCP put all the communication session in one big scatternet. The same reason makes the end-to-end delay in TPSF+ much lower than in BTCP.
**Figure 4.6 Aggregate Throughput of DH3 Packet**

**Figure 4.7 End-to-End Delay of DH3 Packet**
Figure 4.8 Aggregate Throughput of DH5 Packet

Figure 4.9 End-to-End Delay of DH5 Packet
For TCP performance, we used the non-persistent ON/OFF traffic. During the “on” periods, packets are generated at a constant burst rate. During the “off” periods, no traffic is generated. Burst times and idle times follow the exponential distributions. Configuration parameters are as follows: the packet size is 1000 bytes, the average “On” time is 0.5 seconds, the average “Off” time is 0.5 seconds, and the average file size is 1000 packets. The simulation was run for 500 seconds. The metrics used for comparison are the aggregate throughput and the average end-to-end delay. The aggregate throughput is defined as the total throughput obtained by all the TCP connections. The average end-to-end delay is determined from the time the packet is created to the time the packet is received on the application layer. Figures 4.10 - 4.11 show the aggregate throughput and end-to-end delay for TCP, respectively. The end-to-end delay for TCP traffic in TPSF+ fluctuates slightly because TCP may involve congestion control for the packet collisions. With the number of communication sessions increasing, the increase of the aggregate throughput becomes slower because of the increasing packet collisions.

For TCP and UDP traffic, there are more packet collisions when the number of communication sessions increases. The average delay increases due to the packet retransmissions. As expected, when compared with BTCP, the on-demand scatternet in TPSF+ achieves a higher aggregate throughput and a lower end-to-end delay for both UDP and TCP traffic even with the packet collisions.

4.4.2 Average Hop Distance, Connection Delay and Path Successful Ratio with Variant K

With a large coverage area ranging from 400 meters² to 2400 meters², corresponding to the masters’ distance from 1 to 6, three performance metrics for the on-demand scatternet are evaluated: (1) Average hop distance - the average number of hops in the on-demand scatternet created by TPSF+; (2) On-demand scatternet connection delay -
Chapter 4 Simulation Model and Performance

Figure 4.10 TCP Throughput Performance

Figure 4.11 TCP End-to-End Delay Performance
Figure 4.12 On-demand Scatternet Connection Delay

Figure 4.13 Successful Path Connection Ratio among Slaves
this includes the duration of the on-demand scatternet route discovery and reverse path setup; (3) Successful path connection ratio among slaves - the ratio of the number of successful connected paths to the total number of communication paths requested.

If the number of slaves per piconet is 10, Figures 4.12 - 4.13 illustrate the on-demand scatternet connection delay and successful path connection ratio respectively, with variant \( K \), where \( K \) is the maximum intra-piconet distance in the number of hops. From these results, it can be shown that the successful path connection ratio is smaller with a smaller \( K \) because it needs a longer path in one piconet if the node density is low. The average on-demand scatternet connection delay is calculated by the time the source node initiates the on-demand scatternet route discovery until the on-demand scatternet is created. In the simulation, we assume that the timeout value is 20 s when the on-demand scatternet route discovery is regarded as unsuccessful by the source node. The source node will then initiate an on-demand scatternet formation among the master and bridge nodes of the control scatternet. In Figure 4.13, when the masters’ distance is 5, we notice that the connection delay with \( K \) equal to be 3 is much higher than the delay with \( K \) unlimited because of the increasing number of on-demand scatternet route discovery failures.

Figures 4.14 - 4.16 show the average hop distance with the number of slaves per piconet equal to 10, 20, and 30, respectively. By restricting the maximum intra-piconet distance \( K \), we can reduce the on-demand the average hop distance in the on-demand scatternet. From these results, it can be shown that the average hop distance can be reduced by restricting the maximum intra-piconet distance with high node density.

4.4.3 Successful Path Connection Ratio of TPSF+ Compared with TPSF

For the performance comparison between TPSF and TPSF+, we used the mobility model described in Section 4.2.3, in which each slave is moving within its master’s range with variant maximum speed of 2 m/s, 4 m/s, 6 m/s, 8 m/s and 10 m/s. For each speed, the sim-
Chapter 4 Simulation Model and Performance

10 slaves per piconet

Figure 4.14 Average Hop Distance with 10 Slaves per Piconet

20 slaves per piconet

Figure 4.15 Average Hop Distance with 20 Slaves per Piconet
Chapter 4 Simulation Model and Performance

Figure 4.16 Average Hop Distance with 30 Slaves per Piconet

Figure 4.17 Successful Path Connection Ratio
ulation was run 2000 times. The source node starts searching the on-demand scatternet route at a random time in 10 seconds after it connects to the control scatternet. Figure 4.17 shows that when the node speed increases, the successful path connection ratio is slightly reduced using TPSF+, whereas the successful path connection ratio is reduced greatly using TPSF. In TPSF, the route information is collected when the nodes initially access the network but the route information may not be up-to-date because of the movement of the nodes. In TPSF+, the route information is collected when the communication is required, which increases the chances that an on-demand scatternet is successfully created.

4.5 Summary

In this chapter, we described the simulation model and performance analysis. In comparison with BTCP, TPSF+ achieves a much higher aggregate throughput and lower end-to-end delay. When compared with TPSF, the successful on-demand scatternet connection ratio is much higher in TPSF+ in the case where the slave randomly moves around the master node in the control scatternet. Simulation results also showed that the average hop distance of on-demand scatternet is higher when the number of slaves participating in the on-demand scatternet route discovery increases. By applying the maximum intra-piconet hop distance, the average hop distance of the on-demand scatternet can be reduced if the number of slaves participating in the on-demand scatternet route discovery is large. With the slaves randomly moving around the master node in the control scatternet, TPSF+ achieves a higher successful path connection ratio than TPSF. Therefore in the dynamic environment, TPSF+ is more suitable than TPSF.
Chapter 5 Conclusions

We conclude this thesis with a summary of our contributions and directions for our future work.

5.1 Summary

In this thesis, we analyzed the previous scatternet formation algorithms and presented the challenges of Bluetooth scatternet formation. We proposed a new scheme called TPSF+ in which an alternative on-demand scatternet formation algorithm is designed. The main contributions of this thesis can be summarized as follows:

- The on-demand scatternet route information is discovered on-demand, which makes the scatternet formation algorithm more suitable in a dynamic environment. To show the performance, we used the random waypoint mobility model in our simulation. The results showed that the performance is much better in terms of the successful path connection ratio, compared with the TPSF.

- The maximum intra-piconet distance is applied to make the average hop distance of the on-demand scatternet smaller.

- In TPSF+, a small number of nodes is chosen to participate in the on-demand scatternet route discovery procedure so that the redundant on-demand scatternet route discoveries can be avoided.

- To reduce the control packet overhead, PM_ADDR (8 bits) is used in the on-demand scatternet route discovery instead of BD_ADDR (48 bits).

- In order to be realistic, we created a packet collision model for the performance analysis. When compared with BTCP, TPSF+ achieves a much higher aggregate throughput and lower end-to-end delay for both UDP and TCP traffic.
5.2 Future Work

Some issues need further investigation:

• In this thesis, the packet collisions problem is only considered during data packet transmission. The packet collision problem is not considered during the on-demand scatternet route discovery period. Although the OSRREQ packet is carried by inquiry ID packet and the OSRREQ is small, the packet collisions still exist and it will reduce the successful connection ratio. It remains to be considered in the future studies.

• We did not consider the path maintenance after the on-demand scatternet has been set up and how it affects the throughput and end-to-end delay in this thesis. It remains to be considered in future studies.

• In our scheme, the slaves can randomly move around their master with a high successful on-demand scatternet connection ratio. In future, we need to investigate the scenario where the slaves move between different masters in the control scatternet.

• In our simulation, we used the default scheduling scheme - DDR of the Bluescat simulator [16]. In future, a more efficient scheduling algorithm should be designed for the control scatternet in TPSF+.
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL</td>
<td>Asynchronous Connection-Less</td>
</tr>
<tr>
<td>AM_ADDR</td>
<td>Active Member Address</td>
</tr>
<tr>
<td>ARE</td>
<td>Average Route Error</td>
</tr>
<tr>
<td>ARQ</td>
<td>Automatic Repeat reQuest</td>
</tr>
<tr>
<td>ASP</td>
<td>Average Shortest Path</td>
</tr>
<tr>
<td>BD_ADDR</td>
<td>Bluetooth Device Address</td>
</tr>
<tr>
<td>BTCP</td>
<td>Bluetooth Topology Construction Protocol</td>
</tr>
<tr>
<td>CAC</td>
<td>Channel Access Code</td>
</tr>
<tr>
<td>CBR</td>
<td>Constant Bit Rate</td>
</tr>
<tr>
<td>CID</td>
<td>Channel Identifier</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CSN</td>
<td>Communication Session Number</td>
</tr>
<tr>
<td>DAC</td>
<td>Device Access Code</td>
</tr>
<tr>
<td>DH</td>
<td>Data packet - High Rate</td>
</tr>
<tr>
<td>DM</td>
<td>Data packet - Medium Rate</td>
</tr>
<tr>
<td>DRR</td>
<td>Deficit Round Robin</td>
</tr>
<tr>
<td>EMDST</td>
<td>Extended Minimum Degree Spaning Tree</td>
</tr>
<tr>
<td>FEC</td>
<td>Forward Error Correction</td>
</tr>
<tr>
<td>FHS</td>
<td>Frequency Hopping Synchronization</td>
</tr>
<tr>
<td>GIAC</td>
<td>General Inquiry Access Code</td>
</tr>
<tr>
<td>IAC</td>
<td>Inquiry Access Code</td>
</tr>
<tr>
<td>ID</td>
<td>Identifier</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ISM</td>
<td>Industrial-Scientific-Medical</td>
</tr>
<tr>
<td>L2CAP</td>
<td>Logical Link Control and Adaptation Protocol</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LAP</td>
<td>Lower Address Part</td>
</tr>
<tr>
<td>LMP</td>
<td>Link Management Protocol</td>
</tr>
<tr>
<td>LSF</td>
<td>Loop Scatternet Formation</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control</td>
</tr>
<tr>
<td>MRREP</td>
<td>Masters' Route Request</td>
</tr>
<tr>
<td>MRREQ</td>
<td>Masters' Route Response</td>
</tr>
<tr>
<td>MTF</td>
<td>Maximum Traffic Flow</td>
</tr>
<tr>
<td>NAP</td>
<td>Non-significant Address Part</td>
</tr>
<tr>
<td>OSI</td>
<td>Open System Interconnection</td>
</tr>
<tr>
<td>OSRREQ</td>
<td>On-demand Scatternet Route Request</td>
</tr>
<tr>
<td>PAN</td>
<td>Personal Area Network</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit</td>
</tr>
<tr>
<td>PM_ADDR</td>
<td>Parked Member Address</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>RSSI</td>
<td>Radio Signal Strength Indicator</td>
</tr>
<tr>
<td>SAR</td>
<td>Segmentation and Reassembly</td>
</tr>
<tr>
<td>SCO</td>
<td>Synchronous Connection Oriented</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TPSF</td>
<td>Two-Phase Scatternet Formation</td>
</tr>
<tr>
<td>TSF</td>
<td>Tree Scatternet Formation</td>
</tr>
<tr>
<td>UAP</td>
<td>Upper Address Part</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
</tbody>
</table>
### Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intra-piconet distance</td>
<td>The number of hops that a packet traverses in one piconet within the control scatternet.</td>
</tr>
<tr>
<td>Master’s route</td>
<td>A list of the piconets along the communication path. It is identified by a list of masters’ BD_ADDRs.</td>
</tr>
<tr>
<td>Masters’ distance</td>
<td>The number of intermediate piconets that a packet traverses.</td>
</tr>
<tr>
<td>MRREP</td>
<td>(Masters’ Route Response)</td>
</tr>
<tr>
<td>MRREP packet</td>
<td>MRREP packet is generated by the master of the destination node as the response of the masters’ route request. MRREP packet contains the masters’ route information. Each master can obtain the masters’ route information after receiving it.</td>
</tr>
<tr>
<td>MRREQ</td>
<td>(Masters’ Route Request)</td>
</tr>
<tr>
<td>MRREQ packet</td>
<td>MRREQ packet is used for masters’ route discovery.</td>
</tr>
<tr>
<td>OSRREQ</td>
<td>(On-demand Scatternet Route Request)</td>
</tr>
<tr>
<td>OSRREQ packet</td>
<td>OSRREQ packet is used for on-demand route discovery. The packet is broadcast to the slaves along master’s route, carrying the route information until it reaches the destination node.</td>
</tr>
</tbody>
</table>
Bibliography


[16] Bluescat 0.6, http://www-124.ibm.com/developerworks/oss/cvs/bluehoc/bluescat0.6


[18] ns-2.1b7a, http://www.isi.edu/nsnam/ns/