Scatternet Formation for Multimedia Applications over Bluetooth Personal Area Networks

A Thesis submitted in
partial fulfillments of the requirements
of the degree of

Bachelor and Master of Technology
in
Computer Science and Engineering

Navendu Jain 97400
Department of Computer Science and Engineering

Under the valuable guidance of

Prof. B. N. Jain
Department of Computer Science and Engineering

Indian Institute of Technology, Delhi

2001-2002

Certificate

This is to certify that the thesis titled **"Scatternet Formation for Multimedia Applications over Bluetooth Personal Area Networks"** being submitted by Navendu Jain to the Department of Computer Science and Engineering, Indian Institute of Technology, Delhi, for the award of the degree of Bachelor and Master of Engineering, is a record of bona-fide research work carried out by him under my supervision, and in my opinion, it has reached the standard fulfilling the requirements of the regulations relating to the degree.

The matter and results presented in this thesis are original and have not been submitted to any other institute or university, wholly or in part, for the award of any degree or a diploma.

Prof. B. N. Jain
Department of Computer Science and Engineering
Indian Institute of Technology
Hauz Khas, New Delhi, India

Acknowledgment

I would like to thank my project supervisor, Prof. B. N. Jain, for providing us with invaluable help and guidance throughout the course of this project. We would also like to thank Dr. Rajeev Shorey and Mr. Apurva Kumar, Research Staff Members at IBM India Research lab for their insightful comments and helpful suggestions which helped us to improve our work significantly.

We are also grateful to Mr. S. Negi, Senior Technical Assistant, Cisco Advanced Networking Lab for helping us out will all the technical difficulties in the lab.

Last but not the least, this project would not have been possible without the valuable knowledge and experience gained from the faculty members of Computer Science and Engineering Department. May this project be a contribution to their pain-staking efforts in inculcating the spirit of learning and creating.

Navendu Jain

Contents

List of Figures

Abstract

Bluetooth is a new promising local area wireless technology for pervasive computing designed to enable voice and data communication among various electronic devices. Though not specified in version 1.1 of the Bluetooth specification, communication by way of multihop routing (so characteristic of ad-hoc networks) within a scatternet will offer a new and exciting extension to this technology. And the topology of such an ad-hoc scatternet would have a significant effect on the overall performance of the network. The existing algorithms often become infeasible because they use models where the discovering devices broadcast their Ids and exchange substantial information in the initial stages of the algorithm. Hence the need is to optimize on the topology construction and packet forwarding latency taking into account the limited computing capabilities of the devices.

In this thesis, we present "BTSF" as a novel and practical scheme for building an efficient scatternet and discuss the basic rules followed by the BTSF scheme. This design of the algorithm and application architecture has been targeted towards real-time multimedia specific applications. The algorithm achieves in simplifying both network formation and routing problems while minimizing the number of piconets which a relay node participates as well as the total number of piconets. It is both decentralized and self-healing, in that nodes can join and leave at any time without causing long disruptions in connectivity.

To illustrate this framework, we have developed a bluetooth emulator based on an application architecture for implementation of Multimedia Applications running on the Bluetooth Personal Area Network. In the application framework, various scatternet formation algorithms as well as ad-hoc routing protocols are available as modules which could be loaded depending on the user-requirements.

Finally the effectiveness and performance comparison of the BTSF scheme is evaluated through simulation experiments based on the Bluetooth communication model.

Chapter 1

Introduction

The ubiquitous use of information intensive consumer devices such as cell phones, personal digital assistants (PDAs), and laptop computers have called for a new networking paradigm for interconnecting them. The goal is to create a personal area network (PAN) that accommodates seamless information transfer between different devices with varying capacity in an ad-hoc manner without the need for manual configuration, cables, or wired infrastructure. In December, 1999, an industry consortium known as the **Bluetooth Special Interest Group** standardized Version 1 of a short-range, low-power, RF technology called **Bluetooth** motivated in part by the need for suitable link-layer PAN technologies. Initially, this promising new technology has been used as a WPAN (Wireless Personal Area Network), but now, solutions for point-to-multipoint and multi-hop networking over Bluetooth have started to evolve.

A Bluetooth network is essentially an ad-hoc network which is a collection of wireless mobile nodes dynamically forming a temporary network without the use of existing network infra-structure or centralized administration. Due to the limited transmission range of wireless network interfaces, multiple network hops may be needed for one node to exchange data with another across the network. In such a network, each mobile node operate not only as a host but also as a router, forwarding packets for other mobile nodes in the network, that may not be within the direct reach wireless transmission range of each other. Each node participates in an ad-hoc routing protocol that allows it to discover multi-hop paths through the network to any other node. The idea of an ad-hoc network is sometimes also called an *infrastructure-less networking*, since the mobile nodes in the network dynamically establish routing among themselves to form their own network on the fly. However, in Bluetooth, the wireless nodes are grouped in the form of clusters (piconets in Bluetooth) with cluster-heads facilitating the intra-piconet communication. The clusters are connected to each other through common nodes known as bridges.

The main application scenarios of Bluetooth include Ad-hoc networking and providing a data access point for Internet Access in wireless networks. The well known targeted examples of ad-hoc networks are soldiers using their hand held devices to transfer information over the network, means of communication in disaster recovery operations, students using laptop computers to participate in interactive lectures, business associates sharing information and ideas in a multimedia conference. where both data as well as audio and video transfer takes place between wireless networked hosts.

The Bluetooth communication substrate, consisting of Radio, Baseband, Link Controller and Link Manager, specifies mechanisms for establishing connection with nearby devices in an ad-hoc manner. Unlike traditional wireless LANs which rely on distributed contention resolution mechanisms, Bluetooth is based on a centralized master-slave polling 1.1 scheme known as Time Division Duplex (TDD). Furthermore, Bluetooth achieves robustness against interference from nearby devices by employing a Frequency Hopping Code Division Multiple Access (FH-CDMA) technique.

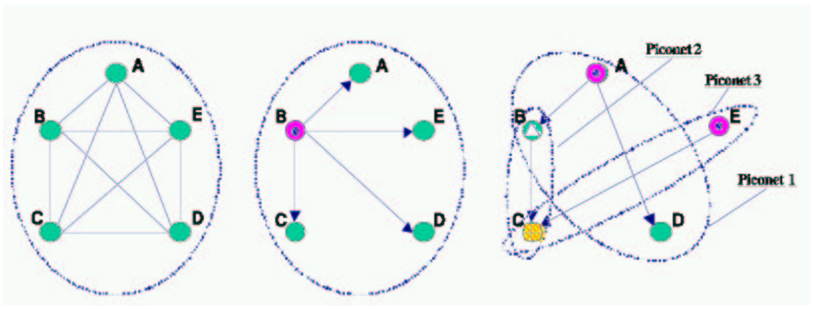

Figure 1.1: (a)Single Channel Model (b)(c) Different Configurations according to the Bluetooth multiple channel model

Because frequency-hopping facilitates high densities of communicating devices, it is possible for dozens of piconets to co-exist and independently communicate in close proximity without significant performance degradation. The Bluetooth specification alludes to this concept of inter-networking multiple piconets, calling it a scatternet [6], but does not specify how it is to be done.

1.1 Objectives

The main objective of this project is to develop a complete architectural framework for providing Multimedia services over the Bluetooth Network adaptable to the dynamic topology changes. This implies adaptation to the changing conditions as well as supporting different kinds of traffic with varying requirements. Further, the scheme should exhibit robustness in adapting to the dynamic network situation. Our aim is to identify these challenges clearly and solve them so that self-organizing Bluetooth scatternets can be realized.
We identify the three main challenges as:

- Study and analysis of different Bluetooth topology formation algorithms.
- Develop a complete architectural framework to support multimedia traffic in dynamic Ad-Hoc networks.
- Design and Implementation of a new Scatternet algorithm and Routing Protocol specifically suited to the Bluetooth specifications.
- Performance Comparison with the existing schemes in the literature.

In broadcast based ad-hoc wireless networks such as *802.11*, the network topology is determined by the physical distance between the nodes. In Bluetooth, an explicit topology formation process is required since nearby devices need to discover each other and explicitly establish a point-to-point link. During the link formation process, the two Bluetooth nodes synchronize the frequency hopping sequence and gather necessary clock information. The essential ad-hoc discovery process could be lengthy and clever solutions are required to quickly form a network topology that spans across all nodes within the transmission proximity.

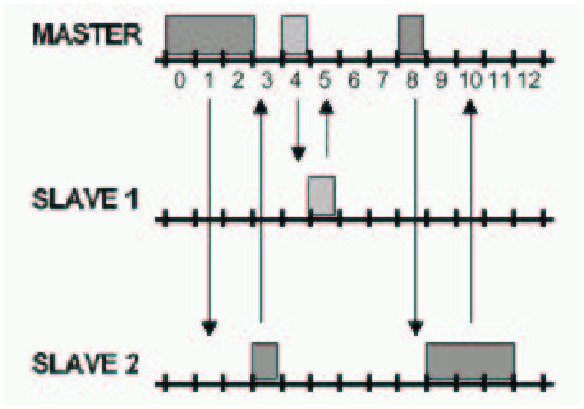

Figure 1.2: Frequency Hopping - Time Division Duplex Bluetooth MAC

A routing mechanism is also essential to route packets over a multi-hop scatternet. Small Bluetooth packet size and low memory and energy requirements dictate the design of *ad-hoc* scatternet routing protocols.

This thesis is organized as follows. In Chapter 2, we specify the bluetooth specifications and characterize the link formation process. We develop some timing analysis for different steps leading to connection establishment. In Chapter 3, a comprehensive survey of centralized and decentralized scatternet algorithms for topology formation and routing is given. In Chapter 4, we move on to the best effort guarantees for multimedia traffic and focus on the complete design and analysis of a distributed algorithm to facilitate the inter-networking between bluetooth devices. Incorporating the specified constraints into the algorithm, we show that our scatternet algorithm is better than the existing algorithms. The complete description of the design and implementation of BTSF algorithm is explained. In Chapter 5, we discuss the development of an architectural framework for applications running on Bluetooth PAN. The effectiveness of our scatternet formation scheme is demonstrated through simulations as well as on real hardware based on the suggested set of evaluation metrics in Chapter 6. Finally, the concluding remarks are given in Chapter 7.

Chapter 2

Bluetooth Specifications

In this chapter, we start the technical specifications of Bluetooth hardware and then describing how two nodes establish a bi-directional communications link. An understanding of this link formation process, which is part of the Bluetooth specification, is necessary to understand topology formation algorithm.

The Bluetooth Baseband Specification [17] defines the Bluetooth point to point connection establishment as a two-step procedure. First neighborhood information is collected through the Inquiry Procedure. The Paging procedure is subsequently used to establish the connections between neighboring devices. Both the Inquiry and Paging procedures are asymmetric processes; they involve two types of nodes (which we call senders and receivers) each performing different actions. During Inquiry, "senders" discover and collect neighborhood information provided by "receivers". During Paging, "senders" connect to "receivers" discovered during a previous inquiry procedure.

2.1 Hardware

The main components of Bluetooth hardware is the radio and the baseband is the wireless link that provides the connectivity between these radios at various devices:

- **Bluetooth Radio:** The Bluetooth radio is a single chip radio - low power device, low voltage RF. and has three standby modes namely *Sniff, Hold, Park.* It allows fast frequency hopping for immediate connection setup and strong interference protection from other devices in its neighborhood. It has a fast ARQ mechanism along-with robust access code and forward header correction.

- **Baseband:** The baseband defines Point to point link of master and slave relationship and radios which can function as either master or slave. This layer runs Time Division Duplex (TDD) protocol between devices in a piconet as well as the connection of relay nodes with their piconets.

2.2 Bluetooth link formation

The link formation process specified in the Bluetooth baseband specification consists of two processes: Inquiry and Page. The goal of the Inquiry process is for a master node to discover the existence of neighboring devices and to collect enough information about the

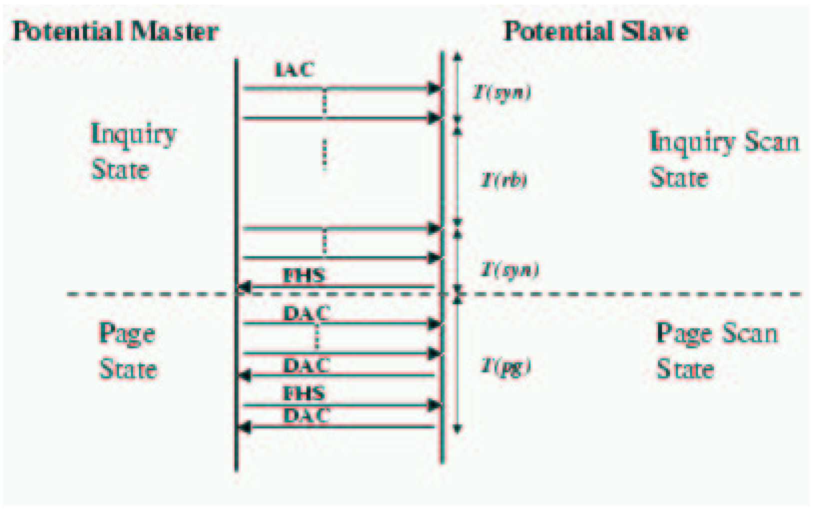

Figure 2.1: Bluetooth Link Formation Process

low-level state of those neighbors (primarily related to their native clocks) to allow it to establish a frequency hopping connection with a subset of those neighbors. The goal of the Page process is to use the information gathered in during the Inquiry process to establish a bi-directional frequency hopping communication channel.

2.2.1 Inquiry Scan

During the Inquiry process, a device enters either the **INQUIRY** or the **INQUIRY SCAN** state. A device in the **INQUIRY** state repeatedly alternates between transmitting short ID packets containing an Inquiry Access Code (IAC) and listening for responses. A device in the **INQUIRY SCAN** state constantly listens for packets from devices in the **INQUIRY** state and responds when appropriate. The Bluetooth specification states that a node in the **INQUIRY** state devotes sufficient amount of time transmitting and listening whereas a node periodically enters the **INQUIRY SCAN** state to scan continuously over a short window.

During the Inquiry process, all nodes hop over 32 dedicated frequencies. Of course, the inquiring node and the scanning node could be out of phase since the phase of each is determined by its local clock. To facilitate proper frequency synchronization within a reasonable amount of time, the Bluetooth Baseband specification requires that the **INQUIRY** node hops at a much faster rate than the **INQUIRY SCAN** node. Multiple **INQUIRY SCAN** nodes can simultaneously receive messages from the same **INQUIRY** node. To avoid contention, each scanning node chooses a random back-off interval, T_{bo}, between 0 and 1023 time slots before responding with the signaling information. If T_{sync} is the delay before two nodes can synchronize their frequencies during the Inquiry process, the time taken to complete the Inquiry process is given by:

$$T_{inq} = 2T_{sync} + T_{bo} \tag{2.1}$$

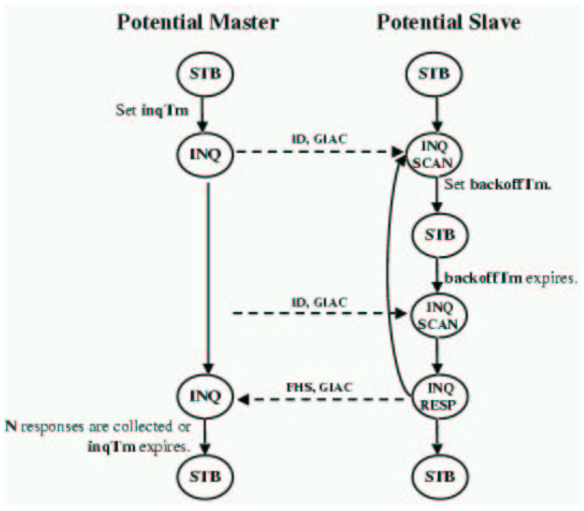

Figure 2.2: State transitions during the INQUIRY process

2.2.2 Page Scan

A node remains in **INQUIRY** state until a timeout period elapses, keeping track of which nodes respond during this time. After this time, if the number of responses is greater than zero, it enters the **PAGE** state. Analogously, a node in the **INQUIRY SCAN** state also periodically enters the **PAGE SCAN** state. A device in the **PAGE** state uses the signaling information obtained during the **INQUIRY** state and sends out trains of ID packets based on the discovered device's address, **BD_ADDR**[1]. When the device in the **PAGE SCAN**

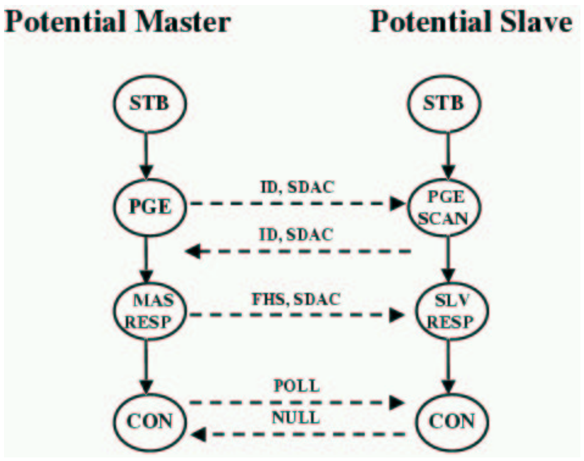

Figure 2.3: State transitions during the PAGE process

state responds back, both devices proceed to exchange necessary information to establish the Master-Slave connection and eventually enter the **CONNECTION** state. The device in the **PAGE** state becomes the master and the device in the **PAGE SCAN** state the slave.

Figures 2.2 and 2.3 illustrate the state transitions during the Inquiry and Page processes respectively. The Page process is similar to the Inquiry process except that the paging device already knows the estimated clock value and **BD_ADDR** of the paged device. However, there will still be some synchronization delay before the pager and the paged devices can communicate. We define T_{pg} as the time taken to complete the Page process. It is worth while to note that it will be most efficient for the two nodes in the Inquiry process to enter the Page process as soon as the inquiring node has received the inquiry response. Thus, the total time taken to establish a link between two nodes is:

$$T_{conn} = T_{inq} + T_{pg} \tag{2.2}$$

T_{inq} is typically much larger than T_{pg} and dominates the delay to enter the **CONNECTION**.

2.3 Network Topology

The basic topology defined by Bluetooth Specifications for inter-connecting devices within range is known as a *piconet*. The devices within a piconet communicate via a single device i.e. they don't have a direct communication channel to other devices in the piconet. *Scatternet* is the next step in forming large infrastructure-less ad-hoc networks for Bluetooth.

2.3.1 Piconet

A piconet is defined as a star shaped topology. The devices assume the role of either a master or a slave. The communication mechanism is centralized in nature as each master periodically polls its slaves for sending and receiving data. A piconet is defined by a unique master and a frequency-hopping sequence which is determined uniquely by BD_ADDR of the master. Each device in the piconet synchronizes to the frequency-hopping pattern of the master device. According to the specifications, the maximum number of active members in a piconet is eight. Since the master can talk to one slave at a time or send a broadcast packet to all of them simultaneously, the piconet supports both Point-to-Point and Point-to-Multipoint connections.

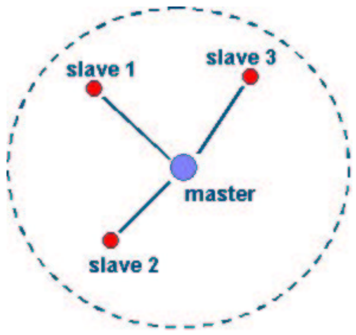

Figure 2.4: Piconet Structure

[1]**BD_ADDR** is the globally unique 48-bit address of the Bluetooth device.

2.3.2 Scatternet

For connecting large number of devices which are also not in range of each other, the logical extension of a piconet is a *scatternet.* A scatternet is a collection of inter-connected piconets so as to facilitate seamless information transfer between any two devices present in the network. The number of devices is expected to be not too large since the interference from neighboring devices is a major constraint in a wireless networks. For this reason, not more than ten piconets are expected to be operating within the same area of range. To address the issue of Inter-Piconet Communication, Bridge nodes are essential to maintain connectivity between their neighboring piconets. These nodes alternatively synchronize and listen to their piconets and pass-on the information from one master to the other. An additional gain of scatternets is the increase in bandwidth since data and multimedia application packets can flow in parallel in both intra as well as inter-piconets.

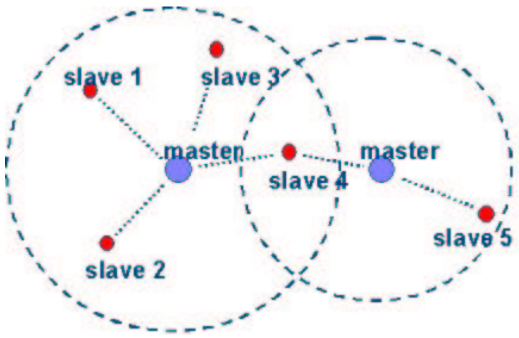

Figure 2.5: Scatternet Structure

2.4 Scatternet Formation

The Bluetooth specification assumes that each node knows whether it is to be a master or a slave. The need for manual configuration of master or slave roles is unattractive when more than a few nodes are attempting to form a connected scatternet in an ad-hoc fashion. To deal with this problem, the Bluetooth specification provides a Host Controller Interface (**HCI**) specification that provides a standardized method of accessing the Bluetooth baseband capabilities. This interface can be used to implement various topology formation schemes.

When a node comes on-line, a potential master node stays in the **INQUIRY** state constantly sending out inquires for the neighboring nodes and attempts to establish links with a potential slaves. A potential slave periodically enters the **INQUIRY SCAN** and the **PAGE SCAN** states and establish links with any master node. Since master nodes always stay in the **INQUIRY** state, it generally follows that slave nodes become bridges between multiple piconets. As time goes on, new links continue to form.

In the next chapter, we address this problem of scatternet formation for supporting real-time applications. The topology formation algorithm executes in three distinct phases starting from independent nodes (denoting connection-less state of the network) culminating in the formation of a scatternet in which master/slave roles have been assigned to each node. The starting phase runs the leader election problem. Once the leader is elected, he starts the process of piconets formation. These piconets in turn join together to form the scatternet network providing full connectivity between any two devices.

Chapter 3

Network Topology Formation Algorithms

Given a collection of Bluetooth devices, an explicit topology construction protocol is needed for forming piconets, assigning slaves to piconets, and interconnecting them via bridges such that the resulting scatternet is connected. Such a protocol should be asynchronous, totally distributed and nodes should start with no information about their surroundings. The problem of constructing distributed self-organizing networks has been addressed in the past, but all the efforts so far were aimed at solving the problem by assuming a single broadcast channel and a CSMA style MAC protocol. This problem is significantly harder for frequency hopping based wireless systems.

3.1 Scatternet Characterization

In [15], the authors identified several characteristics, the combination of which makes scatternets different from previously considered networks. Importantly, Bluetooth links are connection-oriented with low-power link modes. They showed that the unique aspects of the technology require a redesign of the protocol structure for link formation, IP routing, and service discovery. They also suggested an alternative approach where there is a single protocol layer providing a level of indirection within the scope of a scatternet and argued for extensive cross-layer optimizations. Specifically, this allowed *(a)links to be kept active only when absolutely required* and *(b) scatternet-wide floods to be minimized by caching service discovery results at all intermediate nodes.* An approach towards understanding the topological structure of scatternets is studied in [3]. The authors showed that the space of all scatternet topologies is so large that it is computationally infeasible to search the space without proper understanding of its mathematical structure. They identified the mathematical properties of the scatternets and described an technique for enumerating all feasible Bluetooth scatternet topologies giving a lower bound on the *message complexity* of distributed topology construction algorithms. In [1], the authors have presented an approach for scatternet scheduling based on sniff mode based on some modifications in the Bluetooth specifications. Link level fairness was achieved through a slot accounting scheme that reallocated unused bandwidth following the idea of max-min fairness. A comprehensive analysis of various scheduling algorithms has been done by Mario Gerla [8].

3.2 Topology Construction

A topology construction protocol is needed to form piconets and interconnect them via bridges. There exists an extensive literature on distributed protocols for self-configuring networks [11]. Little of it, however, deals with the complications introduced by the master-slave frequency hopping TDD MAC layer used in Bluetooth. Essentially, there are two main scatternet network topologies :

- Tree Topology (Figure 3.1)
  The Tree Scatternet Formation algorithm assigns master/slave roles to nodes while connecting them in a tree structure. The aim is to organize several Bluetooth units in a tree structure, in order to avoid cycles and simplify the routing process. Routing is simplified because there is no need to worry about routing loops and there exists a unique path between any two nodes. All the bridges perform the dual functions of master of their underlying level and slave in their parent's piconet.

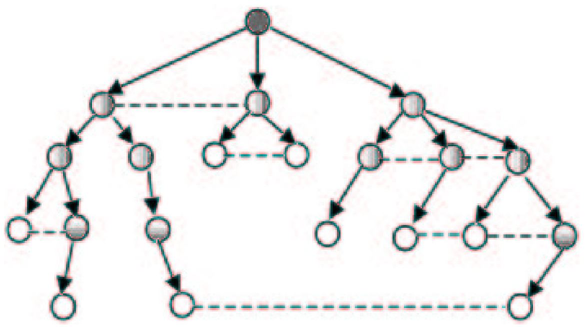

Figure 3.1: Tree Scatternet: Hierarchical Network Formation

- Shared Slaves Topology - SST(Figure 3.2) In this policy, any two piconets share a Slave, i.e., a slave frequently switches between member piconets and thus is active in all of its piconets alternately. Inter-piconet data is routed through the common slaves. This policy is decentralized, more robust and has better load balancing. The shared slave could either be a pure bridge (slave in all member piconets) or a device which alternates being a master and slave in its respective piconets.

Both the above topology formation schemes have their own pros and cons. For the tree based schemes, the topology is the best for networks with frequent broadcasts. Routing is simplified, efficient and has the least-overhead. Though the scheme selects the smallest possible number of links to form a connected scatter and tries to spend the least of network resources on maintaining the scatternet, the resulting scatternet has inherent deficiency due to its hierarchical structure :

a. First, it lacks reliability. If one parent node is lost, all the children and grandchildren nodes below it will be separated from the rest of the network and part of the tree or even the whole tree has to been rebuilt in order to retain the connectivity. In a mobile network, this may happen quite frequently, making the Bluetree very susceptible.

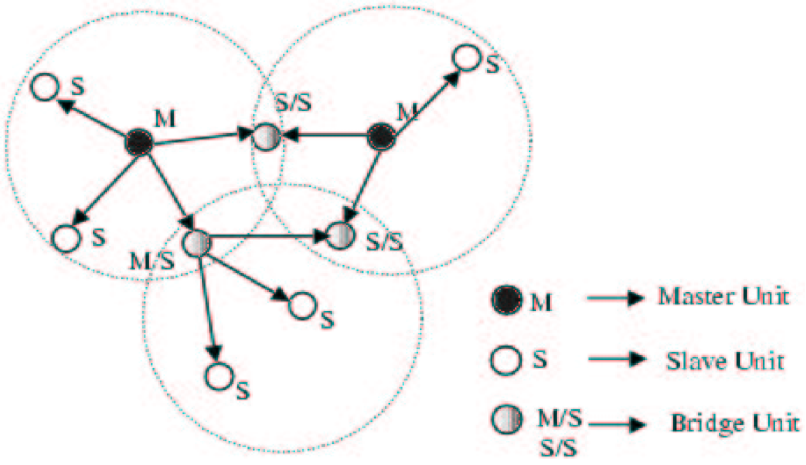

Figure 3.2: Shared Master/Slave Scatternet: Flat Model

b. Second, it lacks efficiency in routing because all the routing paths have to traverse the tree in upward and downward directions. This becomes even worse in a larger system.

c. Third, the parent nodes in Bluetrees are very likely to become communication "bottlenecks" and make it difficult for the network to afford multiple communication pairs. The root node is obviously overburdened under heavy traffic and would end up to be the bottleneck of the entire system.

d. The tree based algorithm does not offer a clear control over the structure of the resulting scatternet, that can result in highly unbalanced trees.

For shared slaves based scatternets, the scatternet construction proceeds in a distributed way; i.e., there is no need to designate any root node and it can be carried out at each node based only on the local knowledge of the node's neighbors. Unlike the hierarchical structure in tree based structures, this is a much flatter structure. The resulting scatternets maintain a degree of connectivity and balanced structure while avoiding wasting network resources on too many redundant links. This topology is also more robust to node and edge failures unlike tree based topologies where a large portion of the network becomes unconnected on even single failures.
However, this protocol has some deficiencies:

a. Routing is complex as we can have routing loops as well as frequent changes in the routes.

b. In general, there may be only one Slave which is shared among all the piconets in vicinity, and thus acts as a router among the piconets. However, having a single Slave for all inter-piconet communication puts heavy load on the Slave.

c. More overhead for spending more resources on maintaining scatternet links.

Comparing these two schemes, the simulation results show that the shared slaves topology can carry far more communication traffic than tree based Bluetooth scatternets. In the following section, we will start by describing a brief description of the algorithms proposed in the literature which address the above issues. Then we propose protocol requirements that each scatternet algorithm should satisfy and then describe the complete details of our algorithm.

3.3 Network Formation Algorithms

In this section, we describe the existing protocols in the literature which aim to establish an efficient topology for bluetooth networks.

3.3.1 Bluetooth Topology Construction Protocol (BTCP)

Pravin Bhagwat et al. presented a symmetric link formation proposed in [16] which addresses the problem of routing messages and the scheduling of communication events. In this scheme, no configuration of potential master or slave roles is necessary. Every node wishing to establish links with other nodes alternates between the **INQUIRY** and **INQUIRY SCAN** states continuously and attempts to connect with another node which is in a different state. The state residence time is randomized. In the Phase I of the three phase algorithm, a a leader election process occurs to configure a particular scatternet topology. The rest of the algorithm is centralized in nature with the leader forming an SST topology. The rest two phases are given in the tables below (Figure 3.3, 3.4). The scheme is limited to scenarios where all nodes arrive over a small window and are within radio proximity of each other. It does not take into account for scenarios where nodes in the scatternet may arbitrarily disappear due to mobility or other constraints such as drained batteries. This scheme also currently limits the maximum number of nodes involved in the scatternet formation to be 36. The authors show that the performance of their scheme under such constraints is reasonably good.

3.3.2 Tree Scatternet Formation (TRF)

Following a parallel approach, Godfrey et. al [19] have described an distributed on-line topology formation algorithm, called TRF (Tree Scatternet Formation) to build scatternets. TRF connects nodes in a tree structure that simplifies packet routing and scheduling. The design allows nodes to arrive and leave arbitrarily, incrementally building the topology and healing partitions when they occur. Simulation results were presented showing that TRF has low routing latency for forwarding packets but with a large topology generation latency. The paper also claims that their protocol by itself does self-healing function, and is able to apply in a scenario, where nodes arrive and leave in an incremental fashion. However, experimental simulations show that scatternet formation delay with 10 to 60 nodes needs a long time delay of 20 to 80 seconds. This kind of long delay will be a problem for this protocol to deal with the above dynamic scenario. The proposed algorithm is quite simple and its performance is compared with that of a probabilistic scheme. The probabilistic scheme is further simpler and does not allow master nodes to bridge traffic between adjacent piconets.

3.3.3 Bluetrees

In [5], G. Zaruba et al. introduce "Bluetrees" as a protocol for forming connected scatternets, which has two variations, namely, Blueroot Grown Bluetree (Centralized) and Distributed Bluetree. The former builds a scatternet starting from some specified node called

```
Input:  Node $L$: the leader of participating Bluetooth Devices, $L \in Nodes$
Output: Role Assignment and Message Transmission to all master nodes.
/* Algorithm running at the leader node $L$ */
  Decide Master/Slave Assignment to each node $n \in Nodes$ ;
  Form a temporary piconet with $L$ as master and designated "$MASTERS$" as slaves;
for-each-node $m \in MASTERS$
     {
          Transmit : $m \longleftarrow$ Connectivity_List($m$);
          Transmit : $m \longleftarrow$ Start Phase III;
     }
Node $L$ breaks the temporary piconet;
Node $L$ enters Phase III as master node;
```

Figure 3.3: Phase II: Algorithm for Bluetooth Scatternet Formation Protocol

.

```
Input:  Set of Nodes $m \in MASTERS$: the master nodes of piconets formed by Leader $L$
Output: Fully Connected Network: Intra and Inter-Piconet Communication.
/* Algorithm running at each master node */
CoBegin: for-each-node $x \in Slavelist(m)$ U $Bridgelist(m)$
     {
          Page and Connect : $x \longleftarrow$ Info_List($x$);
     }
CoEnd:Form piconet with $m$ as master and designated $x$ as slaves;
Protocol Terminates;
```

Figure 3.4: Phase III: Algorithm for Bluetooth Scatternet Formation Protocol

.

Blueroot, while the latter speeds up the scatternet formation process by selecting more than one root for tree formation and then merging the trees generated by each root. One distinct feature of the Bluetree scheme is that all resulting scatternets assume a topology of spanning tree, where the parent node is master and the children nodes are slaves

3.3.4 Bluenet

A completely distributed shared slaves based topology construction protocol has been discussed in [20]. The algorithm proceeds in three stages starting with unconnected nodes to connected devices in the formed scatternet.

- Phase1:
  Initial piconets formed with some separate Bluetooth nodes left. First all the nodes are void; i.e., no a master and no slave in any piconet. During the inquiry state, every Bluetooth node collects information about its neighbors within radio range; i.e. forms a local visibility graph. Then the Bluetooth nodes enter the page state randomly, trying to invite n (Nmax) of its neighbors to join its future piconet. Once a node

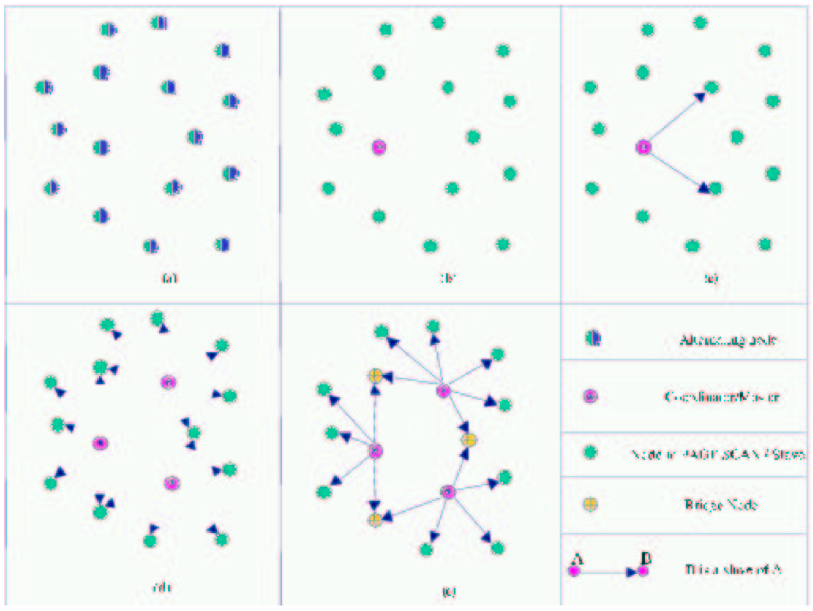

Figure 3.5: Bluetooth Topology Construction Protocol

becomes a slave of some piconet, it will stop paging or answering pages until instructed to begin again by the master. When phase-1 is finished, many separate piconets are formed throughout the system, with some Bluetooth nodes left unconnected to any piconet and all of whose ”neighbors” are associated with a piconet. Some nodes become a master in a piconet and has at most N max slaves. The master then obtains the necessary information about all the slaves in its piconet and broadcasts this information to its slaves. This prevents slave nodes from forming a piconet inside their own piconets later.

- Phase2:
  Separate Bluetooth nodes get connected to initial piconets. During this phase, every separate Bluetooth node begins to page all of its neighbors (but selects at most N max of them to become its slaves eventually) and tries to get connected to some initial piconets built in phase-1. If it gets connected to more than one piconet, it becomes a bridge node. After phase-2, all Bluetooth nodes get associated with at least one piconet.

- Phase3:
  Piconets get connected to form a scatternet. At this point the master of each piconet instructs their slaves to set up outgoing links.

3.3.5 Ching Law

An interesting approach of balanced piconets has been discussed by Ching Law et. al [10]. During the execution of their algorithm, the devices are partitioned into components. A component is a set of interconnected devices. A component can be a single device, a piconet, or a scatternet. There is one leader for each component. For a single-device component,

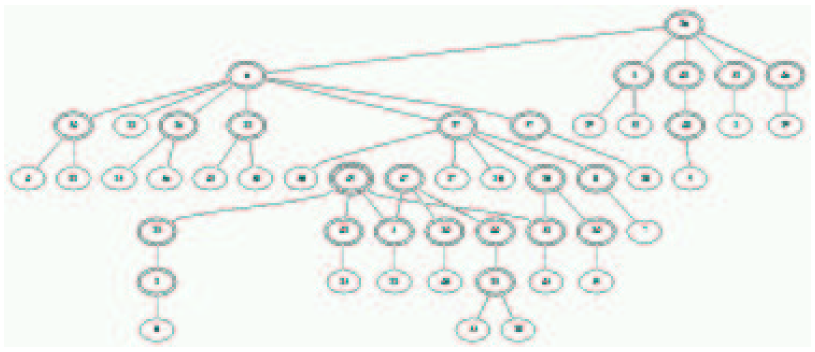

Figure 3.6: Tree Scatternet Formation (TRF)

the only member is the leader. For a piconet, the master is the leader. For a scatternet, one of the masters is the leader. When a leader retires, it stops being a leader and will be inactive for the rest of the scatternet formation algorithm.

Some important invariances for their algorithm are:

- Each leader either has no slave, or has at least one unshared slave in its piconet.
- Each leader has fewer than k slaves in its piconet, i.e., $|S(u)| < k$ for any leader u.

All leaders execute a main procedure in the beginning of each round. Initially all devices are leaders. In procedure main, a leader calls Seek with probability p, $1/3 < p < 2/3$. Otherwise, the leader calls Scan or asks an unshared slave to call Scan. During each round, only one device of each component will call Seek or Scan.

When a leader executes Seek, it tries to acquire a new slave (which is running Scan). However, the leader may not always succeed, because, in any given round, the number of devices running Scan can be fewer than the number of devices running Seek. Therefore, if a leader is not able to contact a slave after certain time, it should give up and run Main again in the next round. Similarly, Scan might also fail in any given round. Essentially, during each round, a matching is found between the Seek devices and Scan devices. The number of connections made (size of the matching) is the smaller of the number of Seek devices and the number of Scan devices. When a leader u running Seek connects to a slave v running Scan, procedure Connected(u, v) is called. If v's other master is w, the piconets of u and w will try to merge if possible. Essentially, if the piconets of u and w can be fit into a single piconet (with at most $k - 1$ slaves), then w and the devices in $S(w)$ become slaves of u. This is performed by the procedure Merge. Otherwise, some slaves are moved from $S(u)$ to $S(w)$ by a migrate procedure. Some other cases also arise finally leading to evenly distributed network topology.

3.3.6 Clustering Algorithm

In this algorithm [14], the nodes elect multiple Masters autonomously to determine potential Masters. This algorithm also requires all nodes to alternate between INQUIRY and INQUIRY.SCAN states, which increases the expected time for discovering a node. It assumes that up to *logS* bits of information can be piggybacked on the Inquiry_response packet. The basic idea of this algorithm is that nodes discovering each other form a tree of responses, the root of each tree being a Master (see Figure 3.7). This parallelizes the formation of each cluster. Each node i maintains a variable *i.phase* which is the number of Inquiry_responses received by it and all the nodes its subtree. Once a node receives an Inquiry_response from another node, it increments its phase by the phase of the replying node. A node which sends an Inquiry_response go to the PAGE_SCAN state, and is out of the competition for becoming a Master. A node whose phase is $S + 1$ declares itself Master, and all the nodes which replied to it (directly or indirectly) and contributed to its phase, are its Slaves. However, the Master do not at first have the Ids and Clocks of all its Slaves. Therefore, it first connects to all those nodes which directly replies to it. Once a connection is established, the Slave sends it's formation about the replies that it had got, to its Master. This type of chaining of message exchanges eventually leads to the Master collecting information about all the nodes in its subtree. It then connects to all its Slaves directly complete the star formation. However, there is a possibility of the phases of two nodes adding up to more than S. In such a situation, the Master who has received the response instructs either some of its Slaves or those of the responding node to go back to INQUIRY state.

The second half of the algorithm involves the election of Super-master among the Masters. To achieve this, we repeat the above algorithm (after the clusters are formed), among the Masters. In this case, the first node which reaches a phase of k becomes the Super-master and conveys this message to all other masters. The bridges could then be decided in a centralized way.

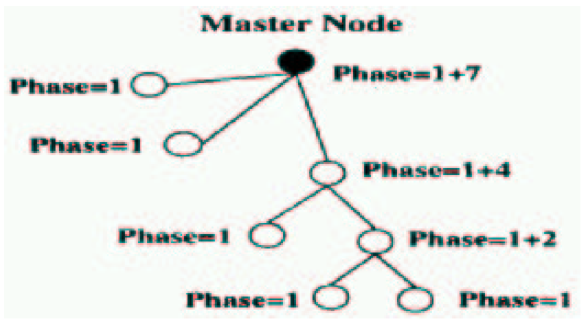

Figure 3.7: Phase I: Clustering Algorithm

Some other interesting research papers are of [2] and [18] from the point of view of theoretical analysis of the scatternet formation problem. In the next chapter, we describe the motivation, constraints and the protocol governing our Bluetooth Scatternet Formation algorithm. We also focus on the routing algorithms for ad-hoc networks in general and list the most significant ones for Bluetooth networks in particular.

Chapter 4

Algorithm Design and Implementation

We have formulated a completely distributed algorithm for scatternet formation and routing for Bluetooth networks. The topology of the formed scatternet is SST. We have also developed a Bluetooth simulator for implementation and comparison analysis of the scatternet formation algorithms and an emulator for the topology formation and routing protocol. In the first part of the project, we did a prototype implementation based on the Linux based Bluetooth emulator, Bluez [9].

4.1 Bluetooth Scatternet Formation Protocol (BTSF)

Our motivation for the scatternet formation problem arises from an infrastructure-less ad-hoc network establishment. Suppose that there are many users in close proximity to each other that wish to form an ad-hoc network using their Bluetooth enabled devices. Each user presses a "start" button and waits for the device to show on the screen a "network connection established" message after a short period of time. After this message appears, the user will be able to exchange information with any other user in his vicinity.

4.1.1 Protocol Features

The description of this application actually contains the elements of a successful connection establishment protocol:

a. Network connection establishment should be performed in a totally distributed fashion. This means that each device starts operating asynchronously on its own and it initially does not have any knowledge about the identities or number of nodes in the room.

b. After completion, the protocol must guarantee a connected scatternet. "Connected" means that there should be at least one path between any two nodes in the network.

c. The network set up delay should be minimized such that it is tolerable by the end user.

d. There should be piconets that have one master and less than seven slaves and that piconets are interconnected through S/S bridge nodes.

In addition to satisfying connectivity, a desirable feature of the protocol would be to be able to shape the network topology according to scatternet formation criteria imposed by specific applications. For example the same node may need to have different roles in different applications. Also it may be possible for a node to have more restrictive degree constraints than seven due to its own nature as a device; for example a palm pilot would not have the processing power to be a master of a seven slave piconet. Scatternet formation criteria could also be in the form of traffic demands that need to be satisfied by the nodes participating in the network construction process. These criteria should be taken into account during the topology construction process if they exist. *The problem of defining scatternet formation criteria is itself an open research issue that is heavily dependent on the envisioned applications.* The link formation protocol is based on a leader election process. Leader election is generally an important tool for breaking symmetry in a distributed system. Since the nodes start asynchronously and without any knowledge of the total number of participating nodes in the network construction process, an elected coordinator will be able to control the network formation and ensure that the resulting topology will satisfy the connectivity requirements of a Bluetooth scatternet.

In the absence of any scatternet formation criteria, and in order to design a simpler and faster protocol, we propose these properties that the resulting network will satisfy:

1. **A bridge node may connect only two piconets. (Bridge degree constraint):**A bridge node forwards data from one piconet to another by switching between them in a time division manner. Given that each portable device may have limited processing capabilities, a maximum bridge degree of two relieves a node of being an overloaded crossroad of multiply originated data transfers. Having more than two increases the connectivity between neighboring piconets. There are several advantages of this :

    - Routing is simplified. The bridge node is to receive packet from one master and to send to the other one (if it is not the destination) with absolutely no computation overhead.
    - Since the maximum value for the connectivity degree of the bridge is two, there are no bottlenecks.
    - Associated with the above factors, there is very low computational and communication complexity.

2. **There are no Master-Slave Bridges (Switch constraint)** Only slave-slave bridges are allowed in the algorithm. There is a bandwidth loss since when the devices won't be able to communicate when their master is active (listening) in the piconet for which it is a slave. For this reason, neither master/master or non-master/non-master connections are allowed. The connectivity metric for the BTSF algorithm is as follows : (Table 4.1).

    Due to the maximal number of non-zero entries in the link formation table, a free device is guaranteed faster connection establishment with the existing network. The only basic requirement for link establishment is that the new device should lie with-in the range of at-least one of the nodes already connected to the network.

3. **Two piconets share at most two bridges (Piconet Overlap constraint).** Bi-connectivity makes the network more tolerant to mobility since the connectivity is maintained as long as at-least one of the links is active. This condition is used in

order to provide a means of terminating easily the connection establishment protocol and calculating the minimum number of piconets. If two masters later wish to share another bridge between them they can do so by means of a bridge negotiation protocol. This however doesn't imply that once the bridge gets connected to two masters, it stops the procedure of PAGE and PAGE_SCAN because that is the fundamental method of obtaining connectivity which should continue as long as the device is running. In this case, once the bridge gets connected to a third master with which either of it's original parents doesn't share a bridge node, it leaves the master having connectivity to both the new and the previous master (since initially, two bridges between them) and becomes a bridge between new and previous masters. For highly mobile scenarios, more than two shared bridges could be considered.

4. **Balance distribution of network resources** There should be uniform distribution of the entire network resources within the entire scatternet so that there are no bottlenecks. In addition to that, given the number of nodes N, the resulting scatternet should consist of the minimal number of piconets possible, similar to the problem of finding the minimal number of routers in an ad-hoc network.

5. **The resulting scatternet should be fully connected** By full connectivity, we mean that there must exist at-least one route between any two pair of devices. Scatternets are expected to change and be reformed over time. A fully connected scatternet in its initial state provides higher robustness against topology changes. Routing is simplified since every master can reach every other master through a bridge node and every slave can reach everybody else through its own master.

6. **Fault-Tolerant to Mobile Ad-hoc scenarios** The algorithm should be able to perform with graceful degradation when subjected to network failures. In such a condition, connection establishment to new devices should occur at a progressive rate.

Table 4.1: Link Formation Combination: Entries with 0 are not allowed

Master/Slave	Master	Non-Master	Free
Master	0	1	1
Non-Master	1	0	1/0
Free	1	0/1	1

4.2 Initialization

The proposed algorithm has only one phase with multiple rounds which results in the establishment of a connected network topology. During the phase, there is an asynchronous, distributed algorithm running at each network device that will eventually know the count, identities and clocks of all the neighborhood nodes participating in the network construction process. As soon as the devices are powered up after initialization, each node starts alternating between the INQUIRY and INQUIRY SCAN state. Any two nodes x and y that discover each other will form a point to point connection, enter a "one-to-one confrontation" and form a connection based on the present status as governed by the algorithm. If both the nodes are unconnected, the device having the larger bluetooth address becomes the winner of the confrontation. Without loss of generality, suppose that x is the winner and y is the loser. For the general case when all the devices may not be with-in each other's range and the slave limit $N_max$of each master is filled, there arises a need for faster connection establishment with new devices in the visibility graph. Due to this reason, our protocol allows the connection of a non-master to a free device. After a sufficient number of connection formation and break-ups with the same free device, a slave device establishes a new piconet with the free device as the master and itself acting a bridge between the new and original piconets. This has the effect of eliminating the free node from the fruitless connection joining process as the network capacities have exhausted, and preparing it for increased connectivity of the entire network.

4.3 Merging of Piconets

Merging of Piconets through agreement between masters. The special case of all-in-range devices leads to an interesting situation when the status of a slave node changes to the bridge node. A simple agreement resolution protocol occurs by which both masters decide if their individual piconets could be combined into a single one. If this is possible, the master having lesser number of slaves transmits the network and clock ids of its slaves and itself. It also tells all its slaves to go into the PAGE_SCAN mode. In turn, the master with larger number of slaves sends a PAGE message to all the slaves of the other master as well as the master itself. The careful reader should note that it is not only possible for the case when all the devices can hear to each other. A possible extension of this useful feature of the protocol for the general case could occur when the agreement also takes into consideration the visibility graph of the masters.

4.4 Generated Network Topology

A step-by-step establishment of the scatternet by BTSF algorithm proceeds in the following way illustrated through an example in Figures 4.1 and 4.2. The first one shows the visibility graph network and the second one displays the scatternet formed by our algorithm.

4.5 Tree Formation Algorithm

In parallel to the BTSF algorithm, we have also developed a tree scatternet formation algorithm for use in our application modules. For this algorithm, there are no specific constraints except those that keep the structure as a tree.

Table 4.2: Link Formation Combination: Entries with 0 are not allowed

Master/Slave	Root	Non-Root	Free
Root	1	0	1
Non-Root	0	0	1
Free	1	1	1

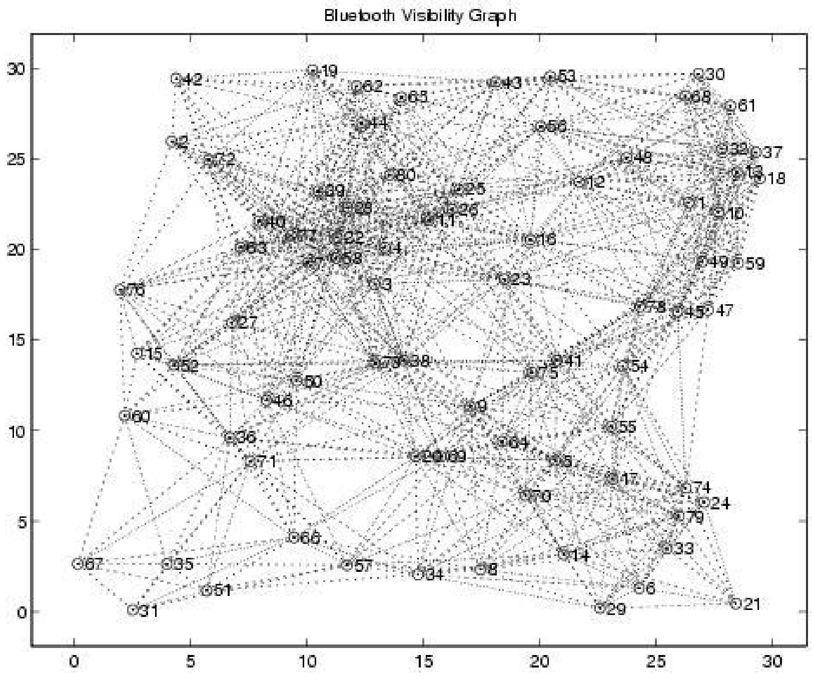

Figure 4.1: Connectivity Network Graph

4.6 Routing

Due to the dynamic nature of Ad-hoc networks, the network topology changes much more frequently than in wired networks. The routing policies for wired networks are not well suited for these kinds of networks. Further the idea of providing the multimedia services requires some quality services, which forces us to look in detail into the routing policies.

A lot of work has already been done in the area of uni-cast routing in ad-hoc networks. These routing protocols can be broadly classified into two categories:

4.6.1 Table driven

Table driven routing protocols attempt to maintain consistent, up-to-date routing information from each node to every other node in the network. These protocols require each node to maintain one or more tables to store routing information and they respond to changes in network topology by propagating updates throughout the network in order to maintain a consistent network view. The areas in which they differ are the number of necessary routing related tables and the methods by which changes in network structure are broadcast. Some examples of table driven protocols are DSDV, CGSR, WRP.

4.6.2 Source initiated

A different approach from table driven routing is source initiated on demand routing. This type of routing creates routes only when desired by the source node. When a node requires a route to a destination, it initiates a route discovery process within the network. This process is completed once a route is found or all possible route permutations have been examined.

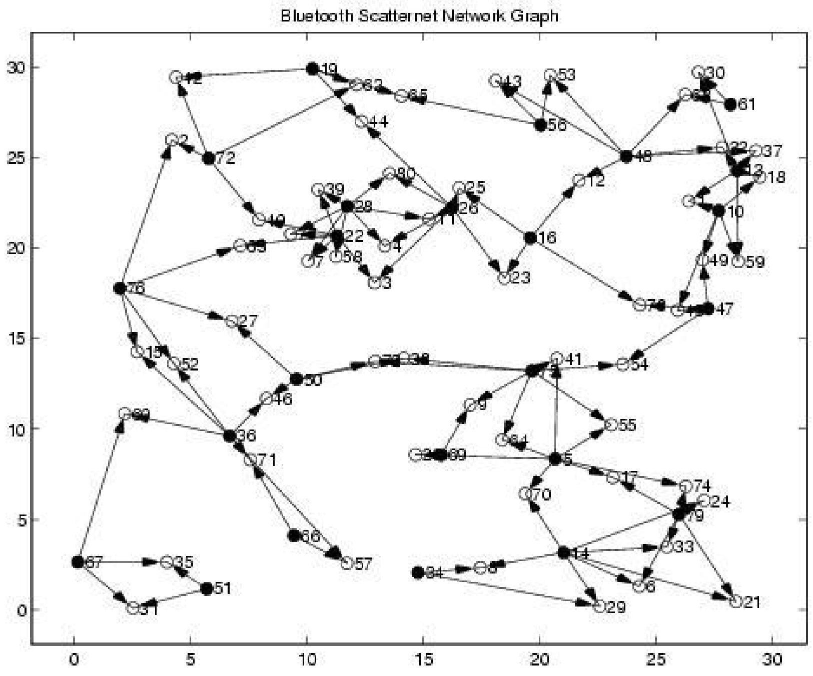

Figure 4.2: Bluetooth Scatternet formed by BTSF

Once a route has been established, it is maintained by a route maintenance procedure until either the destination becomes inaccessible along every path from the source or until the route is no longer desired. Examples are AODV, DSR, TORA, ABR, ZRP, SSR.

A preliminary work on routing in Bluetooth scatternets has been done in [4]. We focus on CGSR and CBRP since they aim to address routing for cluster based wireless lans.

4.7 Ad-hoc Network Routing Protocols

4.7.1 Dynamic Destination-Sequenced Distance Vector Cluster-head Gateway Switch Routing (DSDV)

The Destination-Sequenced Distance-Vector (DSDV) Routing Algorithm [13] is based on the idea of the classical Bellman-Ford Routing Algorithm with certain improvements. Every mobile station maintains a routing table that lists all available destinations, the number of hops to reach the destination and the sequence number assigned by the destination node. The sequence number is used to distinguish stale routes from new ones and thus avoid the formation of loops. The stations periodically transmit their routing tables to their immediate neighbors. A station also transmits its routing table if a significant change has occurred in its table from the last update sent. So, the update is both time-driven and event-driven. The routing table updates can be sent in two ways:- a "full dump" or an incremental update. A full dump sends the full routing table to the neighbors and could span many packets whereas in an incremental update only those entries from the routing table are sent that has a metric change since the last update and it must fit in a packet. If there is space in the incremental update packet then those entries may be included whose sequence number has changed. When the network is relatively stable, incremental updates are sent to avoid extra traffic and full dump are relatively infrequent. In a fast-changing network, incremental packets can grow big so full dumps will be more frequent. Each route update packet, in addition to the routing table information, also contains a unique sequence number assigned by the transmitter. The route labeled with the highest (i.e. most recent) sequence number is used. If two routes have the same sequence number then the route with the best metric (i.e. shortest route) is used. Based on the past history, the stations estimate the settling time of routes. The stations delay the transmission of a routing update by settling time so as to eliminate those updates that would occur if a better route were found very soon.

4.7.2 Cluster-head Gateway Switch Routing (CGSR)

CGSR 4.3 uses as basis the DSDV Routing algorithm. The mobile nodes are aggregated into clusters and a cluster-head is elected. All nodes that are in the communication range of the cluster-head belong to its cluster. A gateway node is a node that is in the communication range of two or more cluster-heads. In a dynamic network cluster head scheme can cause performance degradation due to frequent cluster-head elections, so CGSR uses a Least Cluster Change (LCC) algorithm. In LCC, cluster-head change occurs only if a change in network causes two cluster-heads to come into one cluster or one of the nodes moves out of the range of all the cluster-heads. The general algorithm works in the following manner. The source of the packet transmits the packet to its cluster-head. From this cluster-head, the packet is sent to the gateway node that connects this cluster-head and the next cluster-head along the route to the destination. The gateway sends it to that cluster-head and so on till the destination cluster-head is reached in this way. The destination cluster-head then transmits the packet to the destination. Figure 3 shows an example of CGSR routing scheme. Each node maintains a cluster member table that has mapping from each node to its respective cluster-head. Each node broadcasts its cluster member table periodically and updates its table after receiving other nodes broadcasts using the DSDV algorithm. In addition, each node also maintains a routing table that determines the next hop to reach the destination cluster. On receiving a packet, a node finds the nearest cluster-head along the route to the destination according to the cluster member table and the routing table. Then it consults its routing table to find the next hop in order to reach the cluster-head selected in step one and transmits the packet to that node.

4.7.3 Cluster Based Routing protocol (CBRP)

In CBRP [7], the nodes are divided into clusters. To form the cluster the following algorithm is used. When a node comes up, it enters the "undecided" state, starts a timer and broadcasts a Hello message. When a cluster-head gets this hello message it responds with a triggered hello message immediately. When the undecided node gets this message it sets its state to "member". If the undecided node times out, then it makes itself the cluster-head if it has bi-directional link to some neighbor otherwise it remains in undecided state and repeats the procedure again. Cluster-heads are changed as infrequently as possible. Each node maintains a neighbor table. For each neighbor, the neighbor table of a node contains the status of the link (uni- or bi-directional) and the state of the neighbor (cluster-head or member). A cluster-head keeps information about the members of its cluster and also maintains a cluster adjacency table that contains information about the neighboring clusters.

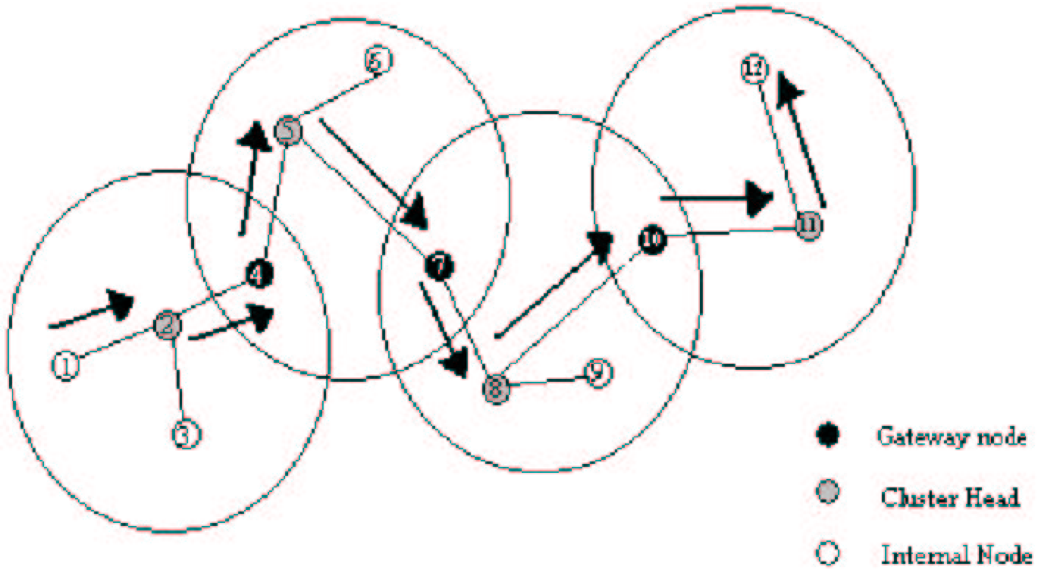

Figure 4.3: Example of CGSR routing from node 1 to node 12

For each neighbor cluster, the table has entry that contains the gateway through which the cluster can be reached and the cluster-head of the cluster. When a source has to send data to destination, it floods route request packets (but only to the neighboring cluster-heads). On receiving the request a cluster-head checks to see if the destination is in its cluster. If yes, then it sends the request directly to the destination else it sends it to all its adjacent cluster-heads. The cluster-heads address is recorded in the packet so a cluster-head discards a request packet that it has already seen. When the destination receives the request packet, it replies back with the route that had been recorded in the request packet. If the source does not receive a reply within a time period, it backs off exponentially before trying to send route request again. In CBRP, routing is done using source routing. It also uses route shortening that is on receiving a source route packet, the node tries to find the farthest node in the route that is its neighbor (this could have happened due to a topology change) and sends the packet to that node thus reducing the route. While forwarding the packet if a node detects a broken link it sends back an error message to the source and then uses local repair mechanism. In local repair mechanism, when a node finds the next hop is unreachable, it checks to see if the next hop can be reached through any of its neighbor or if hop after next hop can be reached through any other neighbor. If any of the two works, the packet can be sent out over the repaired path.

4.8 Scheduling

In [1], the authors have presented an approach for scatternet scheduling based on sniff mode based on some modifications in the Bluetooth specifications. Link level fairness was achieved through a slot accounting scheme that reallocated unused bandwidth following the idea of max-min fairness.

4.9 Salient Features

To conclude this section, the algorithm proposed has the following salient features.

a. **Distributed in Nature**
The algorithm doesn't require any leader election algorithm which is prone to single-point failures. Usually, once is leader is elected, it runs a centralized algorithm to determine the topology and connections of the entire network. Since computational resources of hand-held devices are limited and non-reliable links due to their mobile nature, the scatternet algorithm needs to be completely distributed in nature as well as robust to unreliable nodes and lossy links.

b. **All Devices need not be in radio range of each other**
Most of the existing topology formation algorithms assume that all the mobile devices are within radio-range of each other and can connect to them. This won't be a scenario in many applications where groups of people clustered at different places need intra-group as well as inter-group connectivity ex. to connect to access point. In our algorithm, any device should have a minimum of one device in its neighborhood to connect to the network.

c. **Robustness to Mobility**
Mobility of nodes and lossy links is handled efficiently in our algorithm with its self-healing nature as well as minimum delays sufficient to detect the loss of node or the connecting edge. The algorithm is robust to any transient or permanent failure of the devices without long disruptions in connectivity.

d. **Fast Connection Establishment**
In many application scenarios where devices are switched on simultaneously, a critical requirement is to establish connectivity. There is always a trade-off between efficient network topology and fast connection establishment. This is handled well in BTSF algorithm as would be evident from the scatternet delay formation graphs.

e. **Support for Multimedia Applications**
There is an ever increasing demand for hosting of multimedia and streaming services on mobile devices. In addition to basic applications like chat, file transfer etc., the hot requirements include streaming video and audio. This puts an additional requirement on the network topology for providing multiple paths to the application packets in lieu of the extremely unreliable nature of both nodes and edges. Moreover, there should be a even distribution of resources for all the devices so that no device should become a communication bottleneck for multiple-pair communications.

f. **Minimal Number of Links in a Topology**
Since the mobile wireless devices are always power-constrained, maintaining more than required connectivity would lead to wastage of precious system resources without any given in the network performance since minimal links of links are sufficient for maintenance.

Chapter 5

Application Development

The other main objective of our project is the design and implementation of a complete Application Architecture which would port on any device having Bluetooth Stack.

The motivation for defining such an approach comes from the various usage models envisioned for the technology. The main applicable usage scenarios are as follows :

1. Data and Voice Access Points

2. Establishment of Personal Ad-hoc networks.

3. Cable Replacement Technology

The orientation given by the usage models is that the technology should be ubiquitously deployed irrespective of the hardware or software vendor. It should be easily configured on any hand held portable device which requires connectivity using Bluetooth. There should be a provision of adding value-added services once the basic network is set-up. Users should be able to request for services available on the network and easily use them. New services should be seamlessly added and make known to the running applications at each mobile device. Since large number of devices are expected to be communicating with each other, the application base has to be shifted from a local piconet level to a higher layer of scatternets. In addition to ensuring wider usability of applications, user level transparency of the underlying network should be maintained. Since there are a large variety of devices viz. laptops, palm-tops, cell phones, PDAs etc., the platform independence must be maintained so that various device from different vendors should be able to communicate. The nature of the modules running on such devices should also be of the form of download-plug-n-play.

5.1 Stack and Platform Independence

Related to the above issues, the Bluetooth Stack which forms the link between application level programs and the underlying hardware also needs to be independent.

Bluetooth supports four main generic access profiles viz. Service Discovery Application Profile, TCS-BIN-based profiles, Serial Port profile and Generic Object Exchange Profile. Each of these profiles further include various sub-profiles for running of various applications. Hence the need arises of a layer which interacts with the applications on one end and the profiles in the Bluetooth stack on the other. This standard Bluetooth API would ensure that the underlying application has a completely transparent view of the underlying stack. The API also needs to be augmented with both platform-level and device-level independence. One such API for wide portability for applications targeting Bluetooth has been defined by JSR-82, Java API for Bluetooth wireless technology (JABWT).

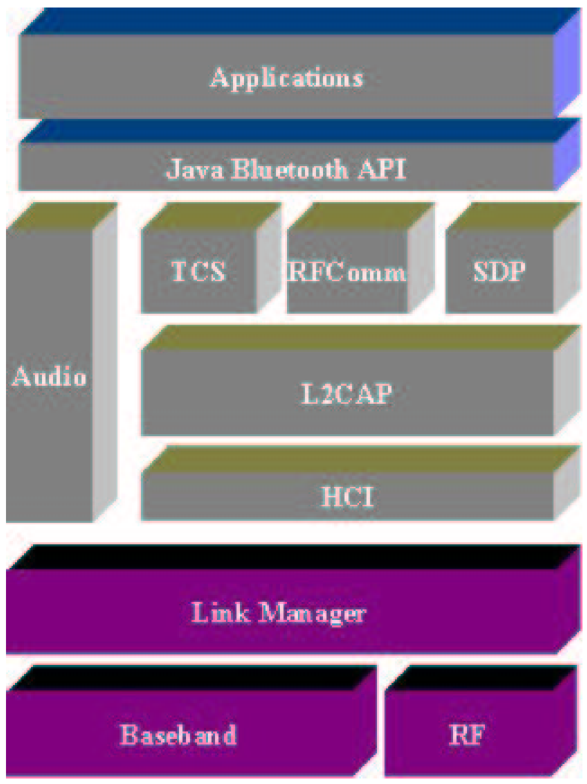

Figure 5.1: Application Architecture over Java API for Bluetooth

Application development using this API has several advantages :

- It is the first industry standard for Bluetooth developed by major industry players. This ensures that the products bundled by these manufacturers would conform to the specifications tightly and support inter-operability between any two devices produced by these companies.

- The Java specification for Bluetooth would inherit the fundamental property of being platform independent from Java itself. Moreover, the specification is built over J2ME, which is the most widely deployed version of Java.

- Java also complements Bluetooth in its own way that it supports various technologies from Core abstraction to service discovery, peer-to-peer ad-hoc networking such as JINI, JavaSpaces and JXTA. So, once the applications are built on top of the JSR-82, they automatically would get bundled with various useful service discovery mechanism to boost the level of services provided by the network.

The Java based API would be implemented mainly in Java and the stack dependent part using JNI Native Interfaces which would communicate to the Stack dependant native code at the lower layer. All of the above modules would exist on the top of the underlying stack which comprises mainly of L2CAP and HCI.

5.2 Architecture

Once the Bluetooth Stack, JSR-82 API and its native code implementation are in place, the next task is to develop the application architecture for running user-level multimedia applications. The architecture envisioned has three main components:

- **User Interface:**
  The application level interface for the user requires J2SE for running swing applications as in our implementation.

- **Scatternet Formation Algorithm:**
  The scatternet formation algorithm is pluggable and portable over J2ME and JSR-82. This implies that various topology formation algorithms would be available as modules to select from and the user would have the choice to pick one from them depending on his specific requirements.

- **Router:**
  The router is also a pluggable and portable module over J2ME and JSR-82 similar to the scatternet formation algorithm. Just like the algorithm, the end-users would have the option of combining the best scatternet algorithm with the best routing algorithm to maximize their system performance.

In order to facilitate the communication flow between these three modules, the well-defined interfaces need to be provided at each end of the module for both the remaining ones so as the underlying implementation and the algorithm of one module should be completely transparent from the point-of-view of other modules. Specifically, the interface between the router and the user interface would provide the mechanism for sending and receiving application packets. Similarly, the interface between scatternet algorithm module and the router would allow the sending and receival of control information and addition, removal of L2CAPConnections between them.

In the first part of the project, we based our design and implementation on the Bluez emulator [9] through which extensive experimentation of various scatternet formation algorithms can be made. The protocol was implemented on top of existing Bluez modules that emulate the Bluetooth environment on a Linux platform. The reason for using an emulator instead of the Bluetooth devices themselves is because current Bluetooth units do not support the piconet switching function and hence cannot operate as bridges. In addition, an emulator provides a higher degree of flexibility, efficiency and modular architecture in testing the system for various parameters and can afford testing the protocol for a large number of nodes. Each Bluetooth host was implemented as a process that mainly consists of two interacting modules. The Bluetooth Baseband (BB) module emulates in software the Inquiry, Paging and piconet switching procedures as defined in the Bluetooth Baseband specification. The protocol module interacts with the BB module through the HCI control specification functions as defined. The use of HCI functions allowed us to later replace the Bluetooth software module with a hardware module, when the bridging capabilities become available in hardware. The wireless medium was simulated by a frequency hopping channel process which is used for the exchange of IAC and FHS packets during the inquiry and paging procedures. The channel process also determines the frequency hopping collisions that are happening between the devices and emulates the FS delays. Note that this channel process is not similar to a CSMA channel since the senders or receivers cannot perform carrier sensing or any kind of intelligent back-off.

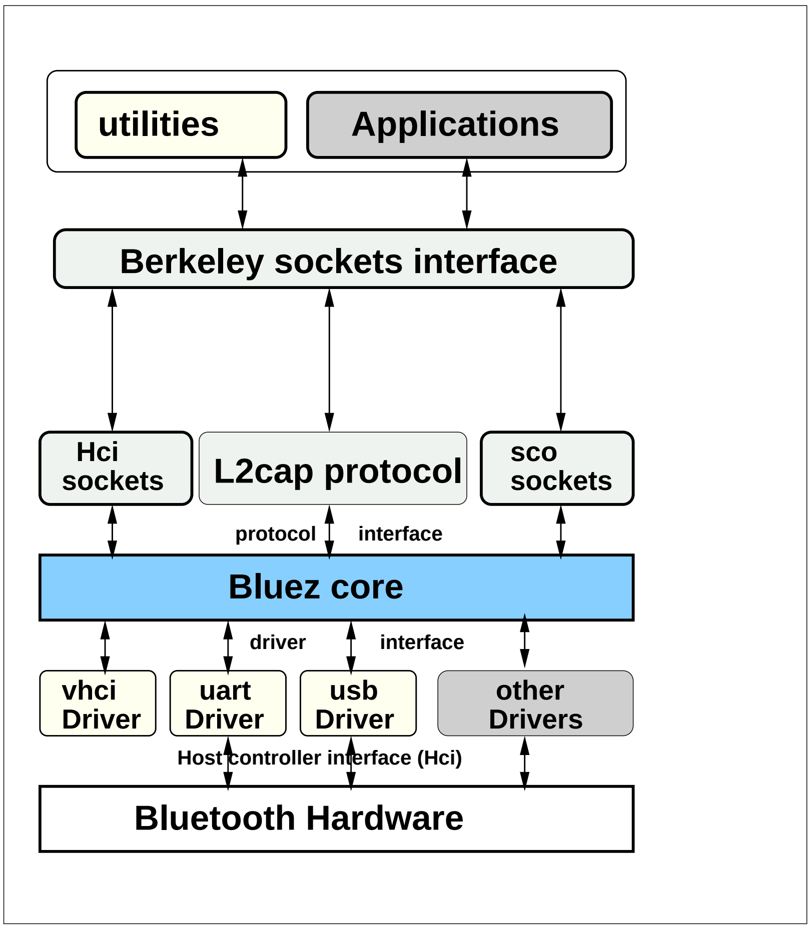

Figure 5.2: Bluez Modules

The architecture of Bluez comprises of Multi-threaded data processing and a standard socket interface to all layers. It can be configured to run on any Linux 2.4.x kernel. It also provides the functionality of hardware abstraction using *Hciemud*, HCI emulator for a Bluetooth device. Currently, Bluez consists of :

a. HCI Core and Connection Scheduler

b. HCI UART, USB and Virtual HCI device drivers

c. L2CAP protocol module

d. Multiple Bluetooth Devices

The detailed diagram of bluez (Figure 5.2) shows all these modules. For emulations involving bluez, we also assume that all the devices are within range of each other. This is a logical assumption for networking many short-range wireless devices in a single room. This fact is mapped in our architecture by having all Bluetooth host processes initially connected to the wireless channel process and executing the topology construction protocol.

5.3 JSR82

JSR82 is the definition of the API's for bluetooth wireless technology for Java 2 Platform Micro Edition (J2ME). This specification is to standardize a set of Java APIs to allow these Java-enabled devices to integrate into a Bluetooth environment. This spec will include basic support for, at least, the following Bluetooth protocols: RFCOMM, OBEX, and Service Discovery protocols. The spec is primarily targeted at native Bluetooth protocols. (There are existing Java IP APIs which can be used to access IP networks from IP enabled Bluetooth devices.)

The Java APIs for Bluetooth are targeted at devices characterized as follows:

1. 512 K minimum total memory available (ROM/Flash and RAM). Application memory requirements are additional.

2. Bluetooth network connection.

3. Compliant implementation of the J2ME Connected Limited Device Configuration.

The specification will define the APIs such that it will be extensible to other Bluetooth protocols which exist today (i.e. Home RF), or that might come about in the future. In addition, the APIs will be specified in a way to allow layering for more capable Java platforms such as the CDC, J2SE, and J2EE. It is envisaged that the Java APIs for Bluetooth will be based on the Generic Connection Framework defined in the J2ME Connected Limited Device Configuration (CLDC) and will use the existing I/O classes of CLDC. This will provide standard Java APIs so that Java applications can be developed for the Bluetooth environment. The need arises since there are currently no standard Java APIs for the Bluetooth protocols.

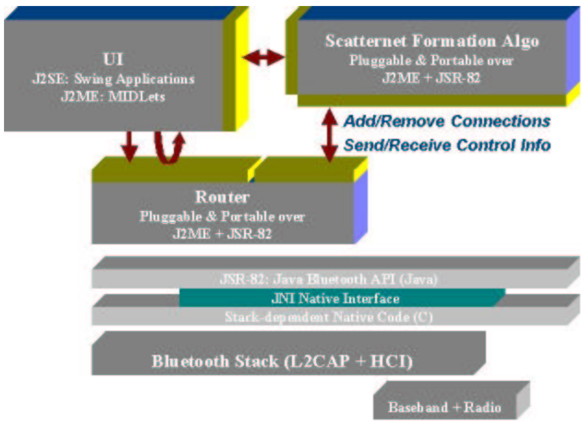
UI
J2SE: Swing Applications
J2ME: MIDLets
Scatternet Formation Algo
Pluggable & Portable over
J2ME + JSR-82
Add/Remove Connections
Send/Receive Control Info
Router
Pluggable & Portable over
J2ME + JSR-82
JSR-82: Java Bluetooth API (Java)
JNI Native Interface
Stack-dependent Native Code (C)
Bluetooth Stack (L2CAP + HCI)
Baseband + Radio

Figure 5.3: Application Architecture

The complete description of the architecture, api and the multimedia application modules can be found in a parallel project thesis of Nitin Pabuwal [12].

Chapter 6

Results and Analysis

We have developed a Matlab based simulator for comparison of topology formation algorithms. We have analyzed them on the following metrics :

1. **Scatternet Formation Latency**
   This is a very important performance criteria since users typically fast connection establishment with the network within some bounds.

2. **Network Diameter**
   Network Diameter is a measure of the longest path between any nodes in the entire network. This directly corresponds to the maximum communication latency or delay a packet would incur traveling between any two points in the entire topology. For a good topology, $O(log(n))$ should be the diameter for n devices.

3. **Number of Piconets**
   Total number of piconets is also an important metric since larger the number of points, longer the end-to-end network delays for packets traversing the piconets and more bridge nodes would be required to maintain the connectivity between them leading to constrain on the resources. A rough estimated of the throughput can also be obtained from the number of piconets. The number of piconets is also equal to the number of masters. Average piconet density can also be computed from this.

4. **Node Capacity (Congestion Level of node)**
   In order to compute the maximum rate at which packets can be pumped into the network, we need to know the capacity function of a scatternet. Since the real capacity-limitations in a scatternet are its node capacities, we simply set all the link capacities equal to $C_0 = 1000Kbps$. The intra piconet overhead is assumed to be $B_1 = 10Kbps$ for each slave that a master node holds; the bridge overhead be $B_2 = 100Kbps$ for each additional piconet that a bridge node joins. Then the capacity of the i^{th} Bluetooth node can be written as:

   $$C_i = C_0 - n_s^i.\triangle B_1 - I_{bridge}^i.(n_p - 1).\triangle B_2,$$

   where n_s^i is the number of its slaves if node-i is a master, otherwise n_s^i is equal to 0; $I_{bridge}^i = 1$ if node-i is a bridge unit and 0 otherwise; and n_p (> 1) is the total number of piconets that a bridge node connects.

5. **Link Coverage**
   Ratio of links used in the scatternet to the number of available communication pairs.

6.1 Algorithm implementation in Application

This section describes the application development based on bluez modules in the first semester of the project. We implemented the algorithm on the emulator demonstrating the topology construction starting from a set of independent nodes in the beginning culminating in the establishment of a fully connected network at the end. We establish the following connections step-by-step during the execution of the protocol:

- *Point-to-Point Connections*
- *Point-to-Multipoint Connections*
- *HCI Sockets Connections*
- *L2CAP Sockets Connections*
- *Scatternet Formation*

The implementation of the protocol starts with the application process opening sockets through the interface provided by Berkeley Sockets Interface (BSD 4.4). This interface connects the higher layers to the one level lower layer consisting of HCI sockets, L2CAP protocol and SCO sockets. Our algorithm of scatternet formation starts with the individual nodes accessing this interface for one-to-one connection establishment with peer nodes when they alternate in **PAGE** and **PAGE SCAN** modes. Further down, the commands go through the main Bluez core. This is the main part of this system which has support for all the lower layer functionalities like UART, USB and other drivers for connecting the radio to the device. The HCI command interface is provided by this layer. When the leader is elected through the distributed leader election algorithm, the leader sends all the encapsulated information through this layered architecture to all the potential masters in the network. These masters in turn have a socket for each of the slaves in their piconet. One important concept to notice is that during the intra-piconet and inter-piconet connection process, the Bridge node is added to its respective piconets only after it has responded back to the masters of these piconets. This ensures that the node is connected to both the piconets so that future routing of packets could be reliably done through this node. Our algorithm not only efficient in the number of piconets also optimizes on the set-up time of the scatternet. The results on the emulation look promising and we plan to port it on the real hardware soon. Some algorithmic improvements have also been proposed which are discussed in the next chapter.

6.2 Scatternet Formation Simulator

In what follows, we examine the suggested scatternet formation scheme by numerical simulations in Matlab. First, n Bluetooth nodes are randomly located within a rectangular area A of size ($X \;\; Y \; meter^2$). The nodes are uniformly placed in A and remain stationery during the simulation. (In this project, we confine our study to stationery Bluetooth networks and will deal with mobility in the future research.) Then the visibility graph network V can be determined according to the Bluetooth radio coverage R. Throughout our simulation, a radio range of 10 meters is assumed with 0 dBm transmitter power. Next, scatternets are built by using the " various algorithm schemas. Finally, the performance of resulting scatternets is evaluated and compared based on the metrics introduced in the earlier sections.

Simulations are carried out with a typical set of parameters defined in the table given below. The parameter N_{max} is chosen to be 5 for some algorithms because their simulations show that such a setting will result in best scatternet performance on the average. A typical visibility graph network for a set of 60 Bluetooth nodes, in which the dotted lines represent all the potential radio links has already been illustrated in the previous chapters. Different scatternets can be formed according to either of the schemes. In the resulting scatternet network figure, the solid circles represent master nodes while the empty circles represent slave nodes. All the scatternet links can carry two-way communications. The arrows on them are only to depict the master-slave relation, i.e., pointing from a master to it slaves.

Table 6.1: Parameters of the Bluetooth Network

Parameter	Value	Description
n	40	Total number of Bluetooth nodes
X Y	20 20	Area size (meter meter)
R	10	Radio range (meter)
Nmax	7	Max number of slaves in a piconet

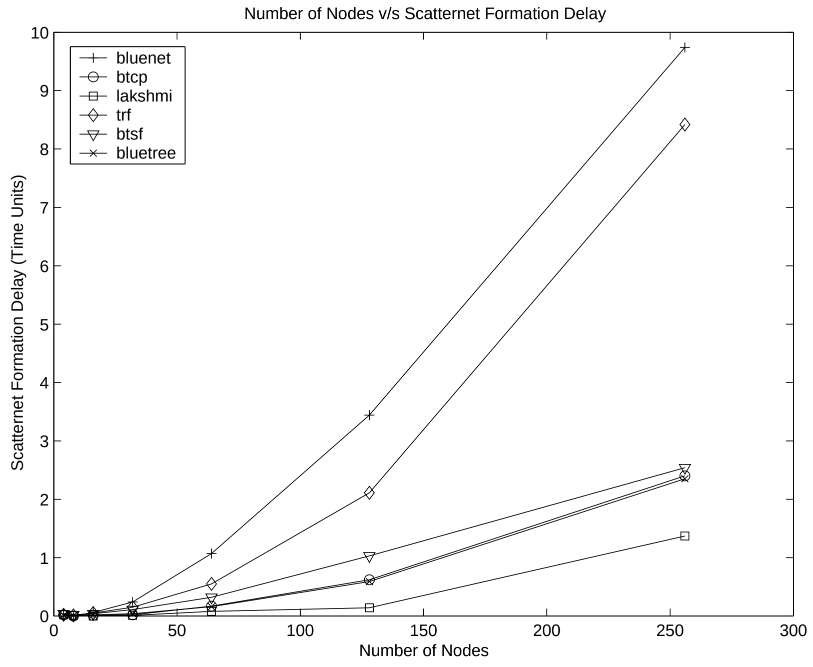

Figure 6.1: Number of Nodes v/s Scatternet Formation Latency

6.3 Performance Assessment and Analysis

The main result from the various plots in this section is that with a little trade-off in the scatternet formation delay, we are able to come-up with the best network topology in terms of number of piconets and the network diameter while still being totally distributed. A main point to note in all these graphs is that no other algorithm is able to beat our performance

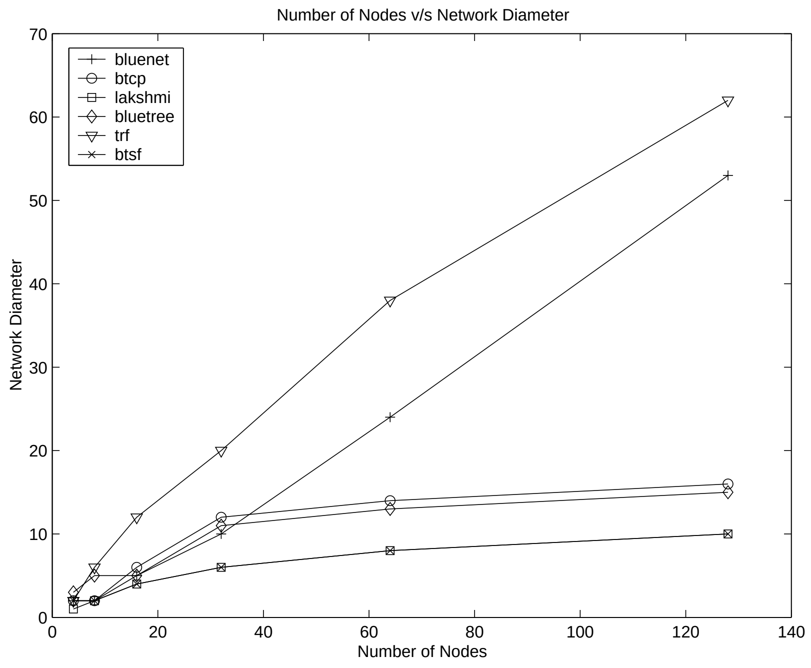

Figure 6.2: Number of Nodes v/s Network Diameter

or even match ours in any more than two graphs, which in our opinion is a significant contribution and strong basis of our algorithm.

6.3.1 Network Establishment Delay

The first comparison is for the scatternet formation latency. Our scheme is close to the minimal time taking algorithm. The minimum time taking scheme has only the major time consuming part of leader election since after that, the entire network formation process is done by the master. Since in our algorithm, there is no single-entity deciding the entire topology i.e. it is completely distributed, we have a small trade-off of consuming little more time than the centralized algorithms. Still, our performance far exceeds those of distributed tree formation algorithms.

6.3.2 Network Diameter

Given the original distribution of the devices, we compute the network diameter of the resulting scatternet structure Graph 6.2. The results exhibit that the resulting network diameter of the graph equals the best algorithm and is in-fact equal to the optimal one. This demonstrates that the BTSF is the most efficient with respect to shortest-routing hops.

6.3.3 Number of Piconets

The number of piconets is also a useful performance parameter for measuring the effectiveness of the scatternet formation algorithm. This is directly linked to the network diameter since more the number of piconets, more would be the end-to-end delay experienced in

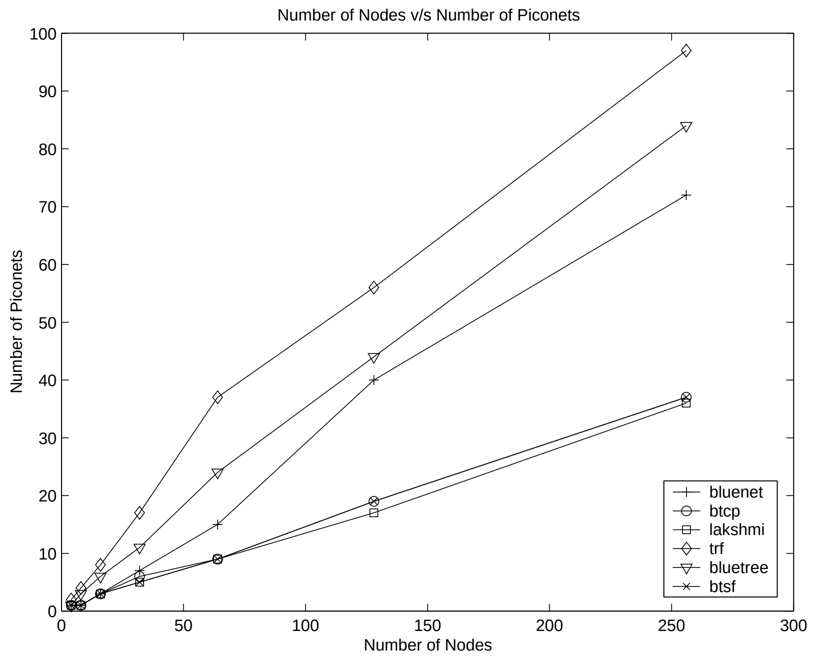

Figure 6.3: Number of Nodes v/s Number of Piconets

reaching the most far-off devices. The results are shown in Graph 6.3. In this case also, our BTSF algorithm is able to out-perform all the other algorithms.

6.3.4 Congestion Level: Mean Node Capacity

As explained in the earlier section, the congestion level of a node describes the available bandwidth or capacity at which a node can transfer i.e. send or receive data to/from the network. The two parameters linked with this are the mean and the deviation. Since we try to form more than one connection with neighboring piconets, we use more network resources which turn out to be only a small difference with the algorithm yielding the highest mean node capacity. This is because we use both the bridges with neighbors to enhance the connectivity of the network. Also, this leads to more robustness against node and edge failures. The results are shown in Graph 6.4.

6.3.5 Congestion Level: Deviation in Node Capacity

Along-with mean node capacity, the deviation in the capacities of the nodes measures the uniform distribution of the resources. Our scheme being distributed has this feature inherently built-in. It is evinced by only the centralized schemes getting ahead since a single point can allocate the resources to the entire network in the best possible and uniform way. The results are shown in Graph 6.6.

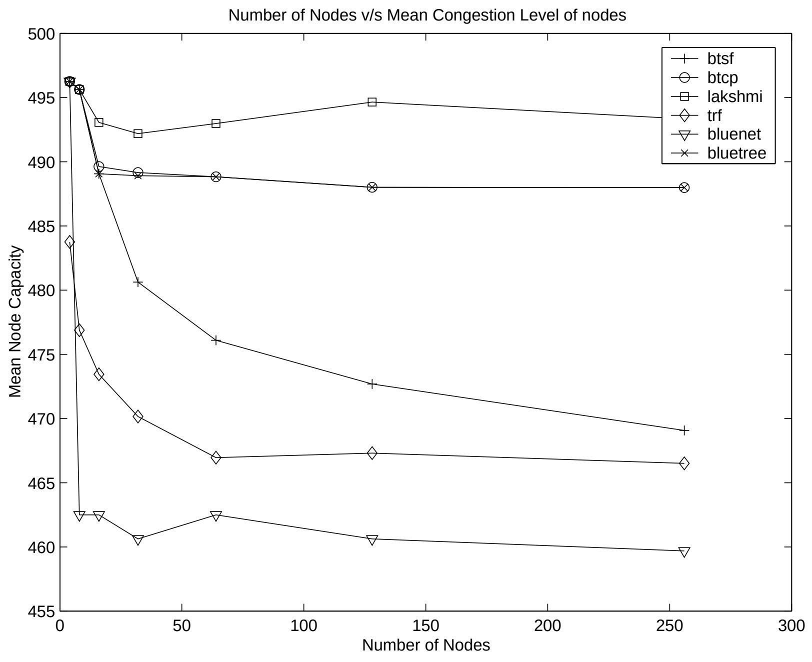

Figure 6.4: Number of Nodes v/s Mean Node Capacity

6.3.6 Link Coverage

Our scheme manages to reduce as much of the bridge overhead as possible while maintaining a good degree of connectivity in the resulting scatternet. As to the performance evaluation of the scatternet, we choose the network diameter and number of piconets to reflect its routing efficiency and to reflect its information-carrying capability. More importantly, the BTSF generated topology is able to carry more communication traffic than any other scheme due to its balanced network topology.

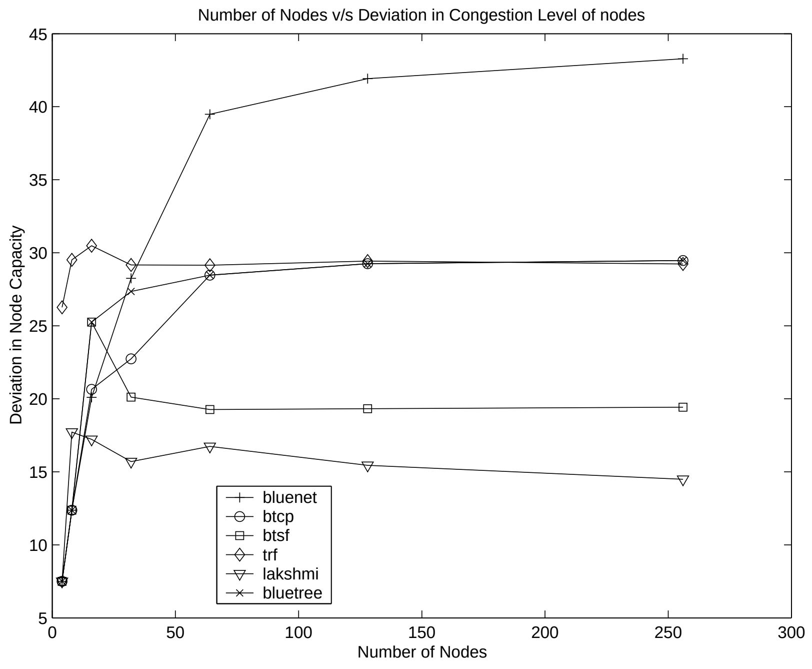

Figure 6.5: Number of Nodes v/s Deviation in Node Capacity

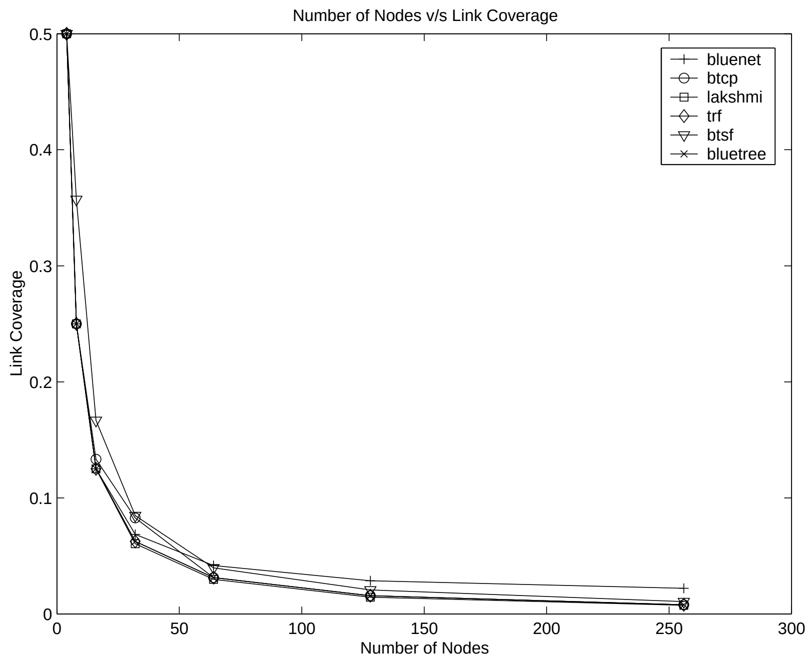

Figure 6.6: Number of Nodes v/s Link Coverage

Chapter 7

Conclusions and Future Work

In this thesis, we have propose a new algorithm BTSF for Bluetooth scatternets formation and maintenance. Our simulations successfully show the relative good connectivity of the scatternet formation. The main highlights of our algorithm are :

- Completely Distributed Network Establishment
- Nearly Optimal number of piconets formed as well as the network diameter
- Network Set-Up Latency Minimal
- Robust to Mobility and maintenance of a connected scatternet at all times.

For the application, we have developed a complete architectural framework for applications running on Bluetooth. It not only meets the basic criteria of connectivity maintenance, but is also :

- Completely Stack, Device and Platform Independent.
- Supports Portable and Pluggable Scatternet Formation and Routing Modules, built over J2ME and JSR-82.
- Provides with a very flexible and robust architecture.

In ad-hoc networks using frequency hopping technology, nodes can be grouped into multiple communication channels. This physical layer setting provides a new way of viewing higher layer functions like topology construction algorithms. Motivated by this environment and using the Bluetooth technology as our research vehicle, we have proposed a mechanism for establishing a connection without any role pre-assignment. Based on the ad-hoc link formation mechanism we presented a distributed topology construction protocol where nodes start asynchronously without any prior neighborhood information and result in a network satisfying the connectivity constraints imposed by the Bluetooth technology. The protocol is centered on a leader election process where a coordinator is elected in a distributed fashion and consequently assigns roles to the rest of the nodes in the system.

This protocol will be tested under a conference-scenario where users arrive in a room and try to form a scatternet by pressing a "button" on their Bluetooth enabled devices. A nice feature of the protocol is that the network formation delay is sub-linear with the number of participating nodes (implying that the users don't need to wait proportionately longer when more users are present). Although, the delay is small, each node must have an estimate of how long it must participate in the protocol before assuming protocol termination. A conservative estimate of the timeout will introduce unnecessary delays in network formation while an aggressive estimate may leave the network disconnected.

7.1 Future Extension

Since our results are both encouraging and promising, the developed framework can be further enhanced by incorporating the following ideas:

a. *Complementing the services with Java-based networking technologies.* This would be a very good value-addition to the applications running over the JSR-82 API. Since Java already supports many of the applications related to service discovery and core abstraction, their incorporation in the Bluetooth based applications would be a very easy task with high-valued gains for the end-user.

b. *An efficient link-scheduling algorithm for bridges* In our algorithm, the bridge nodes periodically hop between all its member piconets in a round-robin fashion. The efficiency of the algorithm can be further increased if we incorporate the hopping sequence of bridges to effectively determine the best sequence of sending control and application packets. A link scheduling mechanism is also necessary since relay nodes need to communicate in multiple channels or links. This is because Bluetooth devices, each with a single radio chip, can only be active on a particular channel at a time and thus, nodes must communicate in different channels on a time division basis (Figure 1.2). Hence, a link scheduling mechanism is required for neighboring nodes to carry out successful packet transfers. In fact, the overall performance of the scatternet-wide communication critically depends on the scheduling mechanism which dictates the bandwidth utilization, end-to-end latency and the energy usage. A scheduling scheme needs to be developed which can adapt to the network changes. Depending upon the network state it would change the time slots distribution with which the bridge nodes participate in their neighboring piconets.

c. *Deploying a variety of applications and algorithm modules over the proposed architecture* Due to their inherent nature of plug-n-play, all these modules would be easily download-able and run on the devices.

d. *Multiple-Route algorithms to in-corporate more than two routes* The proximity of the bluetooth PAN devices provides us multiple paths between any two piconets. Bi-connectivity of the scatternet piconets could be extended to multi-connectivity, which if utilized effectively can lead to a very good multimedia transmission performance.

e. *Commercializing and Continuous up-gradation* Both these things put together would ensure that the technology is able to fulfill the requirements of seamless connectivity at all times regardless of the hardware and software changes, for which it was originally envisioned.

Throughout the design, our aim has been to build a protocol which can be implemented on top of Bluetooth hardware. Although our implementation runs in a Bluetooth emulated environment, when the inter-piconet communication feature is made available in the next release of the Bluetooth hardware, we can test our protocol in an actual setting. We would like to emphasize that this project is the first approach towards tackling the topology construction problem along-with packet forwarding efficiently and providing a fully functional protocol in the Bluetooth frequency hopping environment.

To summarize, we have described an efficient scatternet formation algorithm for networks constructed of devices communicating using Bluetooth. It efficiently connects nodes in a tree structure that simplifies packet routing and scheduling. Unlike earlier approaches addressing this problem, our design does not restrict the number of nodes in the network. It also allows nodes to arrive and leave at arbitrary times, incrementally building the topology and healing partitions when they occur.

Bibliography

[1] S. Baatz, M. Frank, C. Kuhl, P. Martini, and C. Scholz. Adaptive scatternet support for bluetooth using sniff mode. *Proceedings of the 26th Annual Conference on Local Computer Networks, LCN, Tampa, Florida, http://web.informatik.uni-bonn.de/IV/Mitarbeiter/baatz/LCN2001_rp.pdf*, November 2001.

[2] K. Balaji, Sanjiv Kapoor, Amit A. Nanavati, and Lakshmi Ramachandran. Scatternet formation in bluetooth. *IBM Research Technical Report*, 2001.

[3] Pravin Bhagwat and Srinivasa Rao. On the characterization of bluetooth scatternet topologies. *Submitted for Publication, http://winwww.rutgers.edu/pravin/publications/papers/bt-top.ps*, 2002.

[4] Pravin Bhagwat and Adrian Segall. A routing vector method (rvm) for routing in bluetooth scatternets. *Sixth IEEE International Workshop on Mobile Multimedia Communications (MOMUC), San Diego, CA, http://winwww.rutgers.edu/pravin/publications/papers/momuc99.ps*, 1999.

[5] P Gergely, V. Zruba, Stefano Basagni, and Imrich Chlamtac. On the characterization of bluetooth scatternet topologies. *IEEE International Conference on Communications (ICC)*, 2001.

[6] Jaap Haartsen. The bluetooth radio system. *IEEE Personal Communications, Vol. 7(1), Anchorage, AK*, 2000.

[7] Mingliang Jiang, Jinyang Li, and Y. C. Tay. Cluster based routing protocol. *IETF Draft, http://www.ietf.org/proceedings/98dec/slides/manet-cbrp-98dec/, August*, 1999.

[8] Rohit Kapoor, Manthos Kazantzidis, Mario Gerla, and Per Johansson. Multimedia support over bluetooth piconets. *http://www.acm.org/pubs/citations/proceedings/comm/381472/p50-kapoor/*, 2001.

[9] Maxim Krasnyansky. *http://bluez.sourceforge.net*, 2001.

[10] Ching Law and Kai-Yeung Siu. A Bluetooth scatternet formation algorithm. *Proceedings of the IEEE Symposium on Ad Hoc Wireless Networks 2001, San Antonio, Texas, USA, http://perth.mit.edu/ching/pubs/ScatternetAlg.pdf*, 2001.

[11] Nancy A. Lynch. Distributed algorithms. 1996.

[12] Nitin Pabuwal. Scatternet-based multimedia applications over bluetooth personal area networks. *Indian Institute of Technology, Dual Degree Project Major Thesis Report, May*, 2002.

[13] C.E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *Comp. Comm. Rev., October, pp.234-244.*, 1994.

[14] Lakshmi Ramachandran, Manika Kapoor, Abhinanda Sarkar, and Alok Aggarwal. Clustering algorithms for wireless ad hoc networks. *Fourth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Pages 54-63, Boston, MA, August http://www.acm.org/pubs/articles/proceedings/comm/345848/p54-ramachandran/p54-ramachandran.pdf*, 2000.

[15] Bhaskaran Raman, Pravin Bhagwat, and Srinivasa Seshan. Arguments for cross-layer optimizations in bluetooth scatternets. *http://www.cs.berkeley.edu/bhaskar/srouted/srouted.ps*, 1998.

[16] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Distributed topology construction of bluetooth personal area networks. *IEEE INFOCOM http://www.ieee-infocom.org/2001/paper/785.pdf*, 2001.

[17] Bluetooth Special Internet Group Document (SIG). Specifications of the bluetooth system - version 1.1b. *http://www.bluetooth.com*, 2001.

[18] Ivan Stojmenovic. Dominating set based bluetooth scatternet formation with localized maintenance. *Proc. Workshop on Advances in Parallel and Distributed Computational Models, at IEEE Int. Parallel and Distributed Processing Symposium, Fort Lauderdale, http://www.site.uottawa.ca/ivan/07-APDCM-10.ps, April*, 2002.

[19] Godfrey Tan, Allen Miu, John Guttag, and Hari Balakrishnan. Forming scatternets from bluetooth personal area networks. *MIT Technical Report, MIT-LCS-TR-826, http://nms.lcs.mit.edu/projects/blueware/, October*, 2001.

[20] Zhifang Wang, Robert J. Thomas, and Zygmunt Haas. Bluenet - a new scatternet formation scheme. *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02) http://dlib.computer.org/conferen/hicss/1435/pdf/14350061.pdf*, 2002.