A Dynamic and Distributed Scatternet Formation Protocol for Real-life Bluetooth Scatternets

Deepak Jayanna, Gergely V. Záruba
Department of Computer Science and Engineering, The University of Texas at Arlington
deepakjayanna@yahoo.com, zaruba@uta.edu

Abstract

Bluetooth is a universal radio interface for short-range wireless networks. The basic Bluetooth network topology is a single-hop star-shaped *piconet*. Several such *piconet*s can be interconnected into a *scatternet* to form a wireless ad hoc network. This paper proposes a dynamic and distributed protocol to the Bluetooth scatternet formation problem. The protocol was developed strictly within the constraints imposed by the Bluetooth standard, without assuming any outside knowledge on the topology of the underlying connectivity graph or attributes of different nodes. We show relevant performance measures of our scatternet formation protocol by simulations performed with an extended BlueHoc based simulator.

1. Introduction

The massive growth of portable electronic consumer devices such as laptops, cell phones, personal digital assistants (PDA), digital cameras and MP3 players has called for a new way to connect these devices to communicate or share information. Information transfer between these devices, using wires, applying various connectors and protocols has proved to be cumbersome. Recently, Bluetooth (BT) [1,2,3], a new radio interface has been developed to provide device-to-device connectivity via short radio links. BT is a low-cost, low-power, short-range, wireless de-facto standard developed for replacing cables that connect electronic devices.

Most radio systems today are based on a hierarchical structure; base stations are placed at fixed locations that provide local cell coverage, and are connected via a wired backbone infrastructure. Mobile terminals can move within the cells while connected to the fixed network through the base stations. Base stations contain intelligence to provide critical functions like channel selection, registration, etc. On the other hand, in ad hoc systems there is no inherited hierarchical structure, i.e., no distinction is made between base stations and mobile nodes. Ad hoc connectivity is based on peer-to-peer communication between neighbors; there is no wired infrastructure to support connectivity between the mobile nodes. For our purposes we can distinguish between two types of ad hoc networks:

In conventional ad hoc networks, a group of peer units that are all in the transmission range share the same channel: packets transmitted on the wireless channel are received by all nodes within the transmission range. Ad hoc networks based on Wi-Fi (IEEE 802.11x) technology are good representatives for these ad hoc networks.

In a scatter ad hoc network, several groups of units may exist in the same area each with their own logical or real channel. The BT specification enables such scatter ad hoc networking. BT devices need to discover and connect to other devices that are in the vicinity before they can exchange packets among themselves.

We have developed a new scatternet formation protocol that is *distributed* and *dynamic* in nature. The protocol generates a topology that is of the form of a mesh. Master and slave roles are assigned to nodes in a distributed manner thus no centralized decision-making is required. Unlike most of the earlier works our design allows nodes to be added to the network any time. We have extended BlueHoc [11] (a BT simulator by IBM based on ns2) and implemented our proposed scheme; this demonstrates that our scheme can be implemented within the exiting BT specification. Through simulation we show that our scheme achieves connectivity with low delays while trying to minimize the number of piconets.

The remaining sections of this paper are organized as follows. Section 2 gives an introduction to the BT standard. Section 3 describes *scatternets*, i.e., Bluetooth's capability to establish multi-hop networks while outlining some of the more important previous work in scatternet formation. Section 4 is devoted to describing our dynamic and distributed approach to scatternet formation. Section 5 describes our simulation efforts and provides with the performance evaluation of our scheme. Section 6 concludes the paper.

2. Bluetooth Overview

Bluetooth operates in the unlicensed Industrial, Scientific and Medical (ISM) band at 2.4 GHz. A set of

79 frequencies has been defined at 1 MHz spacing. Since the ISM band is open, systems operating in this band need to deal with interfering sources such as microwaves, Wi-Fi devices and baby monitors. Hence, to reduce interference, BT uses frequency hopping spread spectrum (FHSS), with a hop dwell time of 625 μs.

Two or more BT units that share the same channel (same frequency hopping sequence and synchronization) form a *piconet*. One device acts as the master and all other devices act as slaves. Slaves in the piconet can have links only to the master and cannot communicate directly with other slaves, i.e., the master node controls the communication in the piconet. The master also allocates transmission slots to slave nodes. There can be 7 active slaves in the piconet; each slave is given a 3-bit *Active Member Address* (AM_ADDR) so it can be addressed. Full duplex communication is achieved by applying Time-Division Duplexing (TDD).

2.1. Bluetooth Protocol Stack

The BT stack [2] is defined as a series of layers. The Baseband, Link Controller and Link Manager layers form the lower layers and may be implemented on a single chip (BT Module). The Logical Link Control and Adaptation Layer (L2CAP), and RFCOMM and SDP form the upper layers and are usually implemented in software on the host. The Host Controller Interface (HCI) provides a standard serial interface between the lower timing sensitive layers and the higher computational intensive layers. Figure 1 shows the BT protocol stack [2].

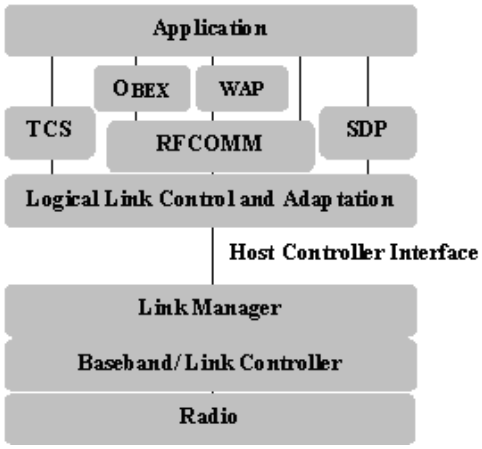

Figure 1. Bluetooth protocol stack.

The radio modulates and demodulates data for transmission and reception on air. The Baseband and Link Controller are responsible for channel encoding, hop sequence selection and synchronization of local and remote clocks. The Link Manager controls and configures links to other devices. The Host Controller Interface handles communication between a separate Host and a BT module. The Logical Link Control and Adaptation layer multiplexes data from higher layers and also handles segmentation and reassembly of packets. RFCOMM can emulate RS232 connection over BT link. WAP and OBEX provides interface to other higher layer Communications Protocols. SDP (Service Discovery Protocol) allows BT unit to discover services offered by other BT devices. TCS (Telephony Control Protocol Specification) provides telephony services.

2.2. Inquiry and Paging

The link formation process in BT consists of two phases: *Inquiry* and *Paging*. During the inquiry phase devices may discover other BT units that are within the range. The discovering device collects the clock and BT address information from devices that respond to the inquiry message. This information is used in the paging phase to establish a bi-directional connection. Figure 2 illustrates the BT link formation process as explained in the next paragraph.

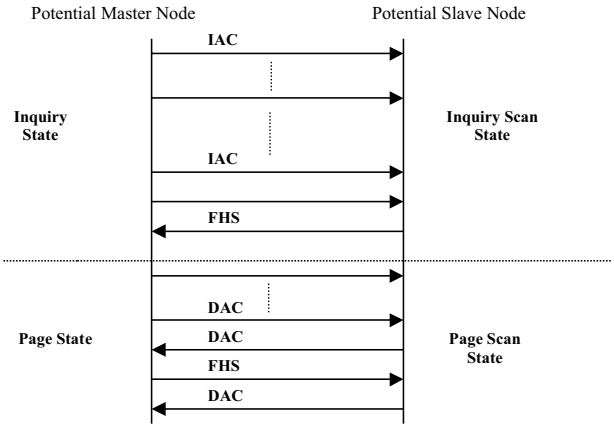

Figure 2. Bluetooth link formation process.

In order for the two devices to discover each other during the inquiry phase, they must be in two complementary substates: *inquiry* and *inquiry scan*. The inquiring device (inquiry substate) sends out inquiry messages (ID packets), which contain the *Inquiry Access Code* (IAC). If another device is listening in the inquiry scan state then a successful inquiry may happen. A device that wants to be discovered enters the inquiry scan state periodically with a period T_{inq_scan} and listens for the duration of the scan window $T_{w\_inq\_scan}$. In order to reduce the discovery time, the inquiring device sends out ID packets at half-slots at two different frequencies and listens at the corresponding receiving frequencies in the next slot. There could be more than one device in the inquiry scan substate listening on the same channel, receiving the same inquiry message. In order to avoid collisions in the reply of such devices, after receiving the inquiry message devices have to choose a random backoff interval (between 0 and 1023 slots) before reentering the inquiry scan state. After reentry and upon receiving an ID

packet the device will respond with an FHS (Frequency Hopping Selection) packet containing its own BT device address and clock.

During the Inquiry phase, both devices follow a 32-carrier frequency inquiry hop sequence. The inquiry hop sequence is divided into two trains: *A* and *B* of 16 frequencies each. A single train is repeated at least $N_{inquiry}$ (256) times before a new train is tried. The BT specification recommends the inquiry substate to last for 10.24 s unless the inquirer collects enough responses and thus elects to abort the inquiry substate earlier.

Paging is similar to the inquiry process but instead of using inquiry access code to capture other devices the paging device uses *Device Access Code* (DAC) (which is derived from slaves BD_ADDR). The paging process can be accelerated because paging device uses an estimated clock value to predict the hop channel the device will start page scan.

3. Ad Hoc Networking with Bluetooth

Multiple piconets may coexist in the same physical space. To interconnect such piconets, BT units can participate in two or more piconets (via time multiplexing) creating topologies called *scatternet*s. Figure 3 depicts a simple Bluetooth scatternet interconnected from 3 piconets. A BT unit can act as slave in several piconets and as a master in only one piconet simultaneously (two piconets with the same master would be synchronized using the same hopping sequence). The interconnection of piconets to form a scatternet is not addressed in the standard and is an open research problem.

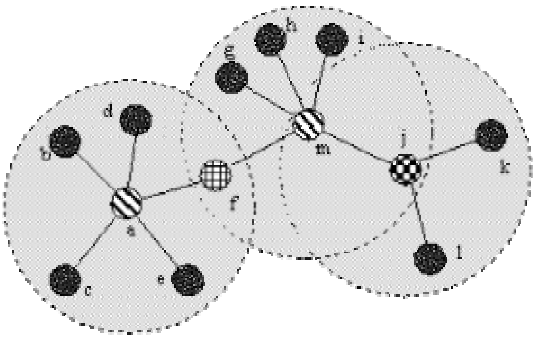

Figure 3. A simple Bluetooth scatternet.

3.1. Bluetooth Scatternet Formation

Some of the important goals for scatternet formation protocols are:

Scalability: The algorithm should work for any number of nodes; not all nodes need to be necessarily in the radio vicinity of every other node.

Low Delay: The time taken to form a connected network should be as little as possible.

Dynamicity and Resilience: Nodes may arrive and/or depart at an arbitrary time.

Decentralized: The algorithm should be completely decentralized. No centralized decision-making.

Minimal number of piconets: The number of piconets in the scatternet should be reduced; by reducing the number of piconets, the average number of hops for messages can be reduced.

Minimal average role for nodes: The number of piconets that a node participates in should be reduced; nodes assuming a large number of roles become bottlenecks.

3.2. Previous Work in Scatternet Formation

Scatternet formation protocols can be divided into single-hop [7,8] and multi-hop [5,6,9] solutions. In the single-hop case all nodes are in each other's radio transmission range, thus restricting their usability. Our proposed solution works for multi-hop scenarios, where not all nodes need to be in each other's transmission range thus forming real ad hoc networks. In the following paragraphs we will show some of the most relevant previous approaches to scatternet formation. Our discussion list is not complete, i.e., it does not show all approaches that have been published but only those that we think are relevant for our further discussions.

Záruba et al. [9] have proposed the first two protocols for forming connected multi-hop scatternets. In both cases the resulting topology is termed a *bluetree*. The number of roles each node takes is limited to two or three. The first protocol is initiated by a given, single node, called the *blueroot*, which will be the root of the *bluetree*. The root node is assigned the role of the master and all of its one-hop neighbors will be its slaves. The children of the root are now assigned an additional master role and all their neighbors that are not assigned any roles will become slaves of the newly created masters. This procedure is repeated recursively until all nodes are assigned roles. In order to limit the number of slaves, the authors observed that if a node has more than five neighbors, then there are at least two nodes among these neighbors that are neighbors themselves. This observation is used to re-configure the bluetree so that each master has less than eight slaves. If a master has more than seven slaves, it selects two of its slaves, *s1* and *s2* that are neighbors themselves and instructs *s2* to be the master of *s1* and disconnects *s2* from itself. In the second protocol, several root nodes are picked and each of them creates it's own scatternet in the first phase of the protocol. In the second phase, the sub-tree scatternets are connected into one scatternet that spans the entire network. The above two solutions are not time efficient. The protocol also assumes all nodes to start the formation process at the same time and that one of the nodes is designated as the blueroot before the formation starts (defeating the decentralized purpose).

In [6] the authors have proposed a multi-hop solution called *BlueMesh*, which defines rules for device discovery, piconet formation and piconet interconnection. The protocol proceeds in two phases (similar to the bluetree approach), *topology discovery phase* and *scatternet formation phase*. In the topology discovery phase each node discovers its one and two hop neighbors (note, that BT does not provide with functions to achieve this). Inquiry procedure in BT does not guarantee a symmetric knowledge of neighbors, in the sense if a node *u* discovers *v*, node *v* may not be aware of *u*. To achieve symmetric knowledge of nodes neighbors the protocol uses inquiry and paging procedures to set up two-node temporary piconets through which two neighboring devices exchange identity. This phase therefore may take several seconds between each node (recall that the recommended time for inquiry is 10.24s). The total time required is not addressed but may be several hundred seconds even for small networks. After having discovered all of its neighbors, a node exchanges this list with its neighbors, thus obtaining two-hop neighborhood knowledge. The scatternet formation phase proceeds in iterations taking care of piconet formation and their interconnection to form a scatternet. The authors [6] do not specify the time taken for the protocol to finish and our view is that this solution is not feasible in a dynamic environment.

Salonidis et al. [7] have proposed another topology construction scheme. In their scheme the nodes alternate between inquiry and inquiry scan state continuously to discover or to be discovered by other nodes. Coordinator nodes are elected, one for each connected component. Coordinators collect information about the whole network, and decide on the roles for each other node. This protocol is limited to single-hop scenarios where all nodes appear at the same time. The authors also assume the number of nodes to be less than 36.

Li et al. [5] have proposed several localized scatternet formation algorithms based on sparse geometric structures. The algorithms have three different phases. In first phase, the neighbor discovery and information exchange phase, the nodes learn about their one-hop and two-hop neighbors. The second phase, planar subgraph construction phase is optional. In this phase, each node computes the incident edges that belong to chosen planar sparse structure, Relative Neighborhood Graph, Gabriel Graph, or Partial Delaunay Triangulation (details of each of these sparse geometric structure is described in [5]). All other edges that are not a part of the sparse structure are removed. The final phase of the algorithm is an iterative phase. The degree of each node is limited to 7 by applying Yao structure [5], and the master-slave relations are assigned in the created subgraphs. The solutions defined by the authors assume that each node knows absolute or relative positions of itself and neighbors (BT does not provide with such functions or the native relay of such information).

Wang et al. [15] have presented a scatternet formation protocol called BlueNet, which has three different phases. In the first phase separate piconets are formed and in the second and third phase the piconets are connected to form a scatternet. Due to the multi-phase nature of this protocol, nodes need to arrive at the same time and start the scatternet formation at the same time.

Law et al. [14] have proposed a randomized distributed scatternet formation protocol and evaluated the performance. The resulting scatternet is a tree, which limits the efficiency and robustness. They also assume that every node is aware of its neighbors and all nodes start the formation process at the same time.

Tan et al. [8] have developed Tree Scatternet Formation (TSF) an online topology formation algorithm that builds scatternet by connecting nodes into a tree structure. Topology produced by TSF at any time is a collection of one or more rooted trees that are autonomously trying to merge to a smaller number of trees. Goal is to form a one rooted tree. Their algorithm is decentralized and self-healing, nodes can join and leave at any time without causing long disruptions in connectivity. However TSF assures connectivity only in single-hop scenarios. Our proposed protocol doesn't have any such limitations.

4. Our Scatternet Formation Approach

In this section we describe our proposed new scatternet formation protocol. The proposed scheme is i) dynamic, i.e., nodes can arrive at an arbitrary time; ii) distributed, i.e., no centralized decision-making is required; and iii) works for multi-hop scenarios. In the previous section we have described some of the limitations of the earlier proposed protocols. Our goal is to propose a new scheme that relaxes some of the limitations of the earlier work and to make it possible to simulate our approach with nothing more but a Bluetooth compliant simulator. Thus, the goals for our scatternet formation algorithm are:

1. To reduce the delay to form a connected topology.
2. To reduce the number of piconets in the scatternet formed.
3. To reduce the average number of roles taken up by a node
4. To be BT compliant, i.e., to consider all constraints imposed by the BT standard (e.g., nodes need not know their absolute or relative location nor the ids of their neighboring nodes in advance).

The communication topology generated by our protocol is a mesh, which has some advantages over the tree structure. The mesh structure is considered to be more robust in an ad hoc environment where nodes arrive and depart at arbitrary times. In the tree structure there is

only one path between every node pair in the connected network. When an intermediate node along the path, which is an internal node in the tree, departs, it could break the tree into several smaller the trees. These trees would have to be merged again to form a fully connected topology. In case of a mesh structure there could be several paths between every node pair and when an intermediate node departs only one of the paths is broken, connectivity is still preserved via other paths. Even though routing decisions are more complex in a mesh structure when compared to a tree structure, packets in a mesh structure can be routed on different paths (e.g., depending on the traffic load).

4.1. Description of Algorithm

In our approach all nodes maintain a *neighbor list*, which can be varied to contain list of nodes in the one-hop, two-hop, etc. piconets. This list is dynamically generated (when a node is not involved in the scatternet this list is empty). We define the one-hop piconet neighbor list to be the *node ids* of all nodes in a piconet(s) that the node participates in. Nodes that are in the adjacent piconet(s) form the two-hop neighbor list (and so on). Nodes that are not a part of any piconet do not have any nodes as neighbors. Table 1 shows a sample 1-, 2-, and 3-hop neighbor set for the scatternet in Figure 3.

Table 1. A sample 1-, 2-, and 3-hop neighbor list.

node	*neighbor_list* one-hop	*neighbor_list* two-hop	*neighbor_list* three-hop
A	{b, c, d, e, f}	{h, i, j, g, m}	{k, l}
B	{a, c, d, e, f}	{h, i, j, g, m}	{k, l}
I	{f, g, h, j, m}	{a, b, c, d, e, k, l}	-
L	{j, k}	{f, g, h, i}	{a, b, c, d, e}

The pseudo-code for the algorithm is shown in Figure 4. A node on arrival picks a role by executing PICK_ROLE procedure. In PICK_ROLE, a node takes either the master role with a probability p, ($p < 0.5$) and calls ROLE_MASTER procedure or takes up the slave role (with probability $1-p$) and calls ROLE_SLAVE. If a node chooses to be a master, then it goes into the INQUIRY state. In inquiry nodes send out inquiry messages trying to discover other nodes that are in the INQUIRY SCAN state. On inquiry timeout (suggested time in BT specification is 10.24 s) or on getting required number of responses the node goes into the PAGE state. In the page state, the master node connects to the newly discovered nodes that do not belong to the neighbor list.

A node does not form new connection with the nodes in the neighbor list to minimize the degree of the piconet (i.e. number of slaves in a piconet) and average number of roles that a node takes. If the neighbor list includes one-hop and two-hop piconet neighbor information then two adjacent piconets can share only one bridge node i.e. only one link connects these two piconets. A node can only have a single unique path with the nodes in the neighbor list, therefore reducing the degree of the piconet and average number of roles taken the nodes.

```
PICK_ROLE ( ) {
        x ← a random number in [0,1]
        if x < p (p < 0.5)
          then MASTER_ROLE ( )
          else SLAVE_ROLE( )
}

MASTER_ROLE ( ) {
        do INQUIRY
        if (new_node discovered & new_node ∉ neighbor_list)
            then form connection to new_node by PAGE
                 neighbor_list ← neighbor_list + new_node.
        if connection complete
            then exchange neighbor information
                 update neighbor_list.
        PICK_ROLE ( )
}

SLAVE_ROLE( ) {
        do INQUIRY SCAN and PAGE SCAN
                if page message & master id ∉ neighbor_list
            then accept connection
       while(slave role timeout)
       if slave role timeout
                PICK_ROLE ( )
}
```

Figure 4. Pseudo-code of algorithm.

The master node - after it finishes paging the nodes that it has discovered - picks a new role with the same probability p. If the node chooses to be a master again and the number of slaves that it has already acquired is less than 7 then it goes into inquiry and paging. In case the node already has acquired 7 slaves it goes into connection, polling the slaves. A timer for master and slave roles is defined, which can be varied depending on the degree and number of roles a node takes. On expiration of the timer the node picks another role again.

4.2. Biasing Inquiry Scan to Inquiry

One of the main contributions of this paper is the consideration of biasing the inquiry scan state to the inquiry state. Intuitively, to ensure a high scatternet throughput, we would like to reduce the number of piconets in the scatternet. This can be done by reorganizing the scatternet after it has been established (e.g., as it has been described in [5,9]). Since each piconet has only one master, another way to reduce the number of piconets is by reducing the number of masters. Such an endeavor may look difficult but considering that nodes that start the inquiry in the inquiry substate are likely to become masters, it is enough to reduce the relative

probability of nodes picking the inquiry substate. Fortunately this probability can be easily changed. If we define the master role probability p to be less than 0.5 then statistically more nodes will take up a slave role. This will also reduce the number of master-slave bridge nodes. As we are going to show in section 5, in a dense environment the p could be reduced to as little as 20%.

5. Simulation and Performance Evaluation

In this section we present our experimental results. To evaluate the performance of our scheme, we have extended BlueHoc [11], a BT Simulator developed by IBM as an extension to the Network Simulator (ns2) [10]. We have implemented the proposed scatternet formation protocol and conducted tests to study the performance.

BlueHoc can simulatee BT piconets, which do not consist more than a master and 1 to 7 slaves. However, BlueHoc lacks support for scatternets. An officially unreleased version of BlueHoc called Bluescat features some support for scatternet scenarios. Bluescat supports scatternet only in a sense that it allows a slave to participate in more than one piconet on a time division basis. We have extended Bluescat to support bridge nodes that take up the roles of master and slave as outlined in [16].

We have considered two different environments; 1) dynamic environment, where nodes arrive at an arbitrary time; 2) static environment, where all nodes start at the same time. In both the cases nodes are uniform randomly scattered over a square area of side L meters. L has been chosen to obtain connected networks. Nodes used are Class 3 BT nodes that have an approximate transmission range of 10 meters. The BT Baseband parameters used for Inquiry and Paging are listed in Table 2. The hold time duration (*holdTO*) used was 2 seconds.

Table 2. Bluetooth baseband parameters in seconds.

T_{inq_scan}	$T_{w\_inq\_scan}$	T_{page_scan}	$T_{w\_page\_scan}$
2.56	11.25×10^{-3}	1.28	11.25×10^{-3}

For the dynamic environment, we have considered two cases for establishing the *neighbor list*. In the first case the neighbor list consists of nodes in the one-hop and two-hop piconets thus named: *2-hop neighbor list case*. In the second case, i.e., the 4-hop neighbor list case, the neighbor list contains nodes that are up to four piconet-hops away. For the static environment, we only considered the 4-hop neighbor list case. We have varied the population from 30 to 60 nodes. We have chosen four different values for probability *p*, for a node to take a master role. The values range from 0.2 to 0.5. All our simulations have been completed with a confidence level above 90% to keep the relative error below 10%.

The metrics used to measure the quality of the scatternet are 1) the average time for the protocol to converge to a steady state, 2) the average number of piconets in the formed scatternet, 3) average number of master-slave role nodes, 4) average number of slave role(s) nodes.

It is difficult to compare our scheme with any of the previous schemes. Due to the lack of simulation tools, Záruba et al. have chosen different metrics to measure the quality of the piconets. The authors of [6] base their comparison on the number of iterations taken for their protocol to obtain a fully connected scatternet. We use the average number of piconets obtained in the static environment case of our scheme to compare with their scheme. Salonidis et al. assume that the clocks of all nodes are synchronized. We do not make any such assumptions. Li et al. apply their algorithm on a sparse geometry structure and make an assumption that every node knows its position (a working simulation model is unavailable). It is difficult to compare their scheme using the metrics we use to evaluate the quality of the scatternet. The scheme described by Tan et al. works only for single hop scenarios and the generated topology is a tree. We therefore chose not to use their scheme for comparison.

5.1. Dynamic Environment

In our experiments for the dynamic environment, half of the nodes arrive randomly over a period of 2 seconds and the other half arrive randomly over a time window of 0 to 20 seconds.

5.1.1. Average Number of Piconets

Figure 5 and Figure 6 show the average number of piconets in the scatternet for the 4-hop and 2-hop *neighbor lists* respectively. The increase in the number of piconets is linear with the increase in the number of nodes and is at its minimum when p=0.2. The average number of piconets for the 4-hop neighbor list is marginally less when compared to the 2-hop neighbor list scenario. This is because in the 4-hop scenario no node shares more than one path between any node that is up to 4 piconet hops away, where as in the 2-hop neighbor list scenario there is no more than one path between nodes that are up to 2 piconet hops away. Nodes in the 4-hop neighbor list scenario cannot form a new link with any nodes that are up to four piconets hops away.

5.1.2. Time to Converge to Steady State

Figure 7 and Figure 8 show the time taken for the protocol to converge to a steady state (achieving >99% connectivity) for the 4-hop and 2-hop *neighbor lists* respectively. Results show that the time is fairly constant with the increase in the number of nodes and in the probability *p*. The time taken by the 4-hop neighbor list scheme to converge is marginally less when compared to 2-hop neighbor list. This is because of the fact that the

number of piconets formed by the 4-hop scheme is less than that of the 2-hop scheme. Therefore the 4-hop scheme reaches a steady state faster.

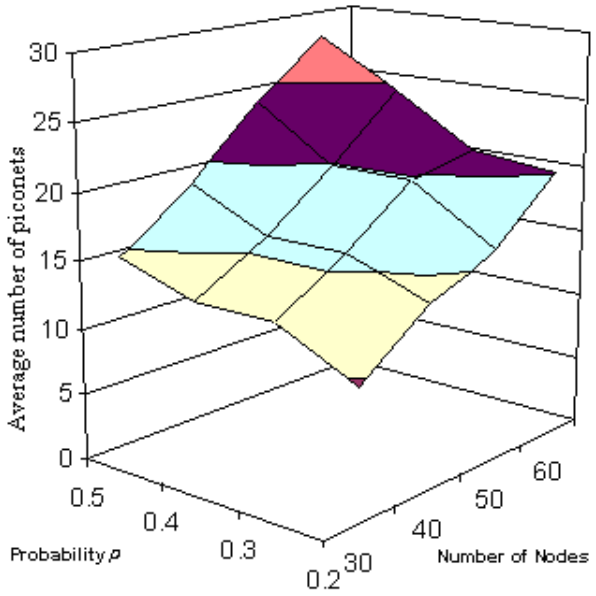

Figure 5. Avg. number of piconets for 4-hop *neighbor list*

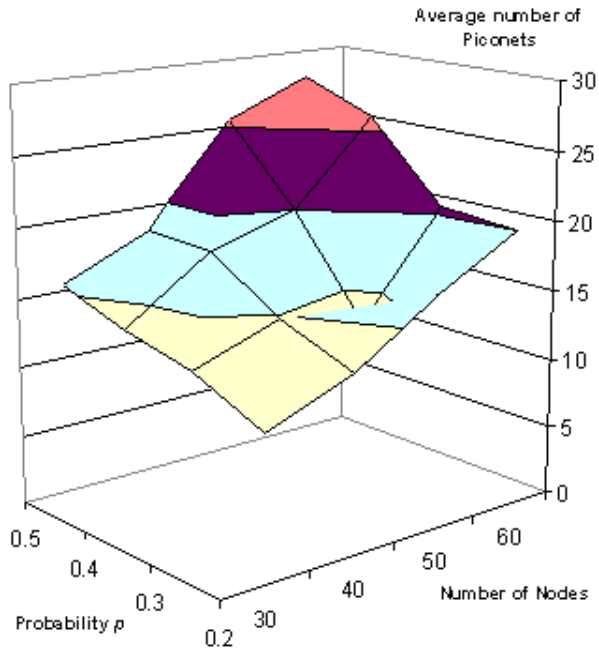

Figure 6. Avg. number of piconets for 2-hop *neighbor list*

5.1.3. Average Number of Master-Slave Role Nodes

Figure 9 and Figure 10 illustrate the average number of master-slave role nodes in the scatternet for the 2-hop and 4-hop *neighbor lists* respectively. The plots show that the average number of master-slave role nodes is less than one when the probability p is 0.2 and increases linearly with the increase in p. The reason for this behavior is due to a reduced number of nodes choosing the master role whenever a node picks a new role.

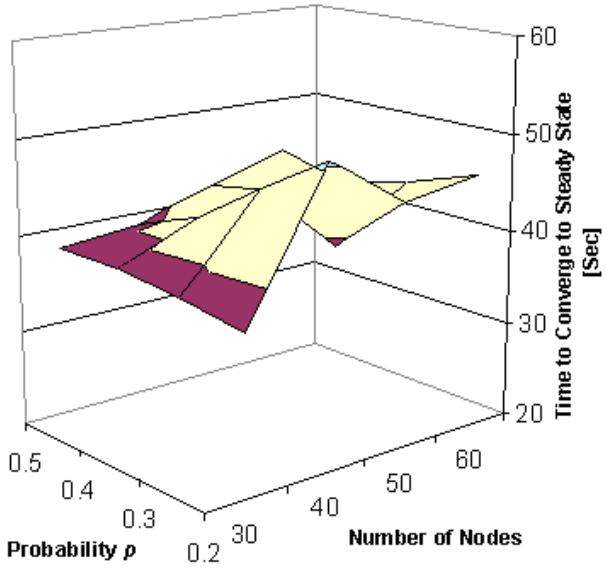

Figure 7. Average time to converge to a steady state for 4-hop *neighbor list*

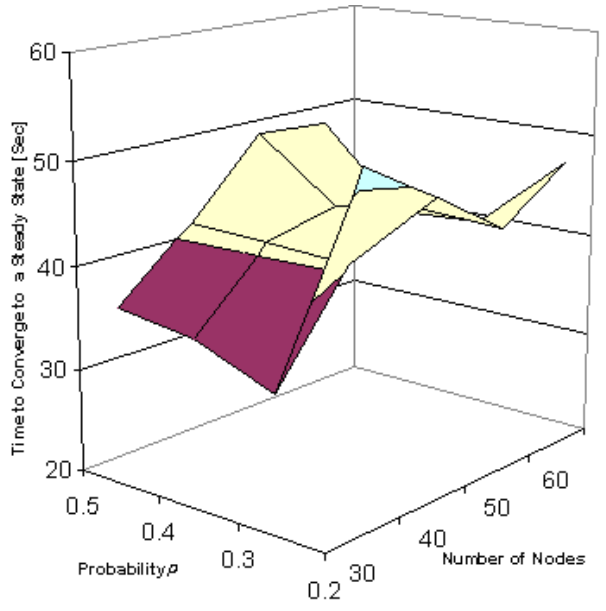

Figure 8. Average time to converge to a steady state for 2-hop *neighbor list*.

5.1.4. Average Number of Slave Nodes

Table 3and Table 4 list the average number of slave nodes that serve as slaves in one or more piconets for the

4-hop and 2-hop *neighbor lists* respectively. S^1 denotes that the node serves as a slave in one piconet and S^2 denotes that the node serves as a slave in two piconets and so on (note that nodes act only as slaves and does not include those nodes that take up master and slave roles).

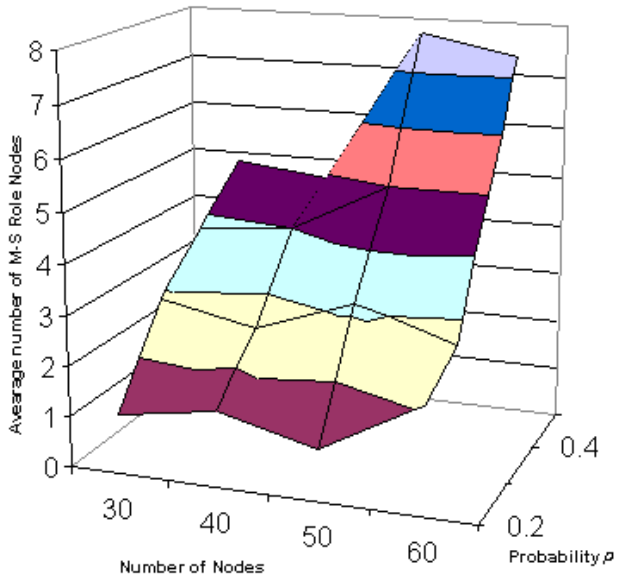

Figure 9. Average number of master-slave role nodes for 4-hop *neighbor list*.

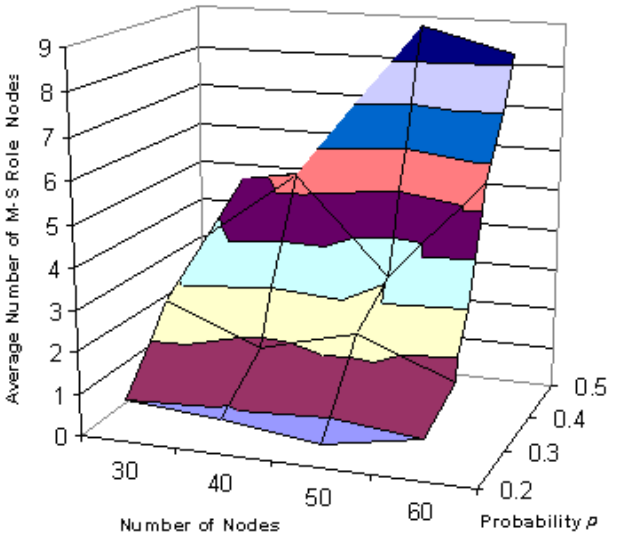

Figure 10. Average number of master-slave role nodes for 4-hop *neighbor list*.

The nodes that take up S^4 and greater were less than 2% of all the nodes and we have therefore omitted them in our tables. Results show that number of S^1 role nodes is highest when $p = 0.2$ and decreases with the increase in p. This behavior counters the behavior of S^3 role nodes, where the numbers decrease with the increase in p; when $p = 0.2$ the number of piconets in the scatternet is the least therefore lesser more number of nodes take up the S^1 role and with the increase in the number of piconets more bridge nodes are required to interconnect the piconets.

5.2. Static Environment

For the static case, nodes start randomly over a small time interval ensuring that clocks are not synchronized.

5.2.1. Average Number of Piconets

Error! Reference source not found. shows the average number of piconets in the scatternet for the 4-hop neighbor list case. The graph is similar to plot obtained in the dynamic environment scenario. The average number of piconets is minimal when p=0.2 and increases linearly with the increase in p.

Figure 11 shows a comparison of our scheme with results of BlueMesh [6]. For comparison we chose the case where $p = 0.2$, since the average number of piconets in the formed scatternet is minimal there. The data points for the BlueMesh scheme were obtained from [6]. Based on our assumptions, our scheme outperforms BlueMesh.

5.2.2. Time to Converge to Steady State

Figure 12 shows the average time to reach steady state with more than 99% connectivity. The behavior seen here is a little different from the dynamic environment case. The time increases marginally with the increase in number of nodes. The reason for this behavior is due to more nodes being active at the same time. In the dynamic case, many nodes arrive when the scatternet formation is already initiated and therefore the time is fairly constant.

6. Conclusions

The inexpensive, short-range, wireless standard Bluetooth can be used to enable ad hoc multihop networking. Multihop topology generation in BT is an open research problem referred to as the scatternet formation problem. In this paper we have proposed a distributed and dynamic scatternet formation algorithm providing a working solution to scatternet formation. Our design does not restrict the number of nodes and works for both single- and multi-hop scenarios not restricting arrival and departure times of nodes. Through simulations we have confirmed that our scheme works within the Bluetooth specification. We have shown the performance of our scheme in measuring the time required for link formation. Study on p (probability that the node takes up the master role) has shown that when p>0.2 the "quality" of the scatternet degrades; since the average number of piconets increases with the increase in p, which results in more number of nodes acting as gateway nodes.

As per our current knowledge our proposed scatternet formation protocol is the only multi-hop solution that can be implemented within the constraints imposed by BT standard.

Table 3. Average number of slave and master nodes for 4-hop *neighbor list*.

# of nodes	S^1				S^2				S^3				Master				M-S			
	Probability *p*				Probability *p*				Probability *p*				Probability *p*				Probability *p*			
	0.2	0.3	0.4	0.5	0.2	0.3	0.4	0.5	0.2	0.3	0.4	0.5	0.2	0.3	0.4	0.5	0.2	0.3	0.4	0.5
30	10.3	7.5	7	5.4	7	5.9	5.9	5.8	2.4	3.7	3.2	3.8	8.4	9.7	8.8	9.7	1.2	2.9	3.8	4.8
40	12.1	9.8	9.2	6.7	8	8.4	8.1	8.4	5.8	5.4	5.3	4.5	12.1	13.4	11.9	14.1	1.5	2.5	4	4.6
50	17.4	9.8	9.3	6.5	11.2	11.8	10.9	10.5	5.2	7.3	7.4	7.2	14.9	16.6	14.8	16.1	1	3.2	5	7.7
60	17.8	17.9	11.5	8.6	14.3	12.9	14.4	11.1	7.9	7.4	8.5	10.9	18.1	18.2	19.6	20.8	2.1	2.6	5	7.3

Table 4. Average number of slave and master nodes for 2-hop *neighbor list*.

# of nodes	S^1				S^2				S^3				Master				M-S			
	Probability *p*				Probability *p*				Probability *p*				Probability *p*				Probability *p*			
	0.2	0.3	0.4	0.5	0.2	0.3	0.4	0.5	0.2	0.3	0.4	0.5	0.2	0.3	0.4	0.5	0.2	0.3	0.4	0.5
30	10.5	8.2	6.2	3.6	6	5.4	5.9	5.9	2.9	4.2	4.3	4.3	9.33	9.9	10.1	10.8	1	2.7	3.7	4.6
40	13.8	10.2	7.1	6.9	8.8	8.4	7.9	8.2	4.2	6.3	6.3	6.2	11.9	13	12.6	13.2	0.8	1.8	5.4	5.1
50	13.3	10.9	9.1	4.3	11.1	11.3	10.9	9.1	7.9	8.2	8.3	9.2	16.2	10.7	16.9	16.7	0.5	2.4	3.1	8.8
60	17.1	16.5	9.3	7.7	14.2	14.7	14.3	13.4	8.1	7.4	9.3	9.7	19.1	19	20.3	19.8	1	1.5	5.6	8.2

References

[1] BLUETOOTH SIG, "Specification of the Bluetooth System, Volume 1: Core, Version 1.1," February 22 2001. http://www.bluetooth.com,

[2] Jennifer Bray, and Charles F. Sturman, "Bluetooth: Connect without Cables," Prentice Hall, 2001.

[3] J.C. Haartsen, "The Bluetooth Radio System," IEEE Personal Communications Magazine, vol. 7, no. 1, pp. 28-36, 2000.

[4] J.C. Haartsen, "Bluetooth – Ad Hoc Networking in an Uncoordinated Environment," Proceedings of ICASS, vol. 4, pp. 2029-2032, 2001.

[5] Xiang-Yang Li, Ivan Stojmenovic, Yu Wang, "Partial Delaunay Triangulation and Degree Limited Localized Bluetooth Multihop Scatternet Formation", to appear in the IEEE Transactions on Parallel and Distributed Systems, 2003.

[6] "Degree-Constrained Multihop Scatternet Formation for Bluetooth Networks," Proceedings of the IEEE GLOBECOM 2002, Taipei, Taiwan, November 2002.

[7] T. Salonidas, P. Bhagwat, L. Tassiulas, and R. Lamaire, "Distributed Topology Construction of Bluetooth Personal Area Networks," Proceedings of the IEEE INFOCOM, 2001.

[8] G. Tan, "Self-organizing Bluetooth Scatternets," Masters Thesis, MIT 2002.

[9] G.V. Záruba, S. Basagni and I. Chlamtac, "Bluetrees-Scatternet Formation to Enable Bluetooth Based Ad Hoc Networks," Proceedings of the IEEE International Conference on Communications (ICC), 2001.

[10] The VINT Project, "The ns Manual," 2002. http://www.isi.edu/nsnam/ns/

[11] "BlueHoc Simulator," 2001. http://oss.software.ibm.com/bluehoc/

[12] J. Chung, and M. Claypool, "NS by Example," Worcester Polytechnic Institute, 2001. http://nile.wpi.edu/NS

[13] Marc Greis et al., "Tutorial for the Network Simulator "ns"," 2002. http://www.isi.edu/nsnam/ns/tutorial/index.html

[14] C. Law, A. K. Mehta, and K.-Y. Siu, "Performance of a New Bluetooth Scatternet Formation Protocol," Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing 2001, October 2001.

[15] Z. Wang, R.J. Thomas, Z. Haas, "Bluenet – a New Scatternet Formation Scheme," Proceedings of HICSS-36, Hawaii, 2002.

[16] Deepak Jayanna, "A Dynamic and Distributed Scatternet Formation Protocol for Real-life Bluetooth Scatternets," Master Thesis, The University of Texas at Arlington, 2003.

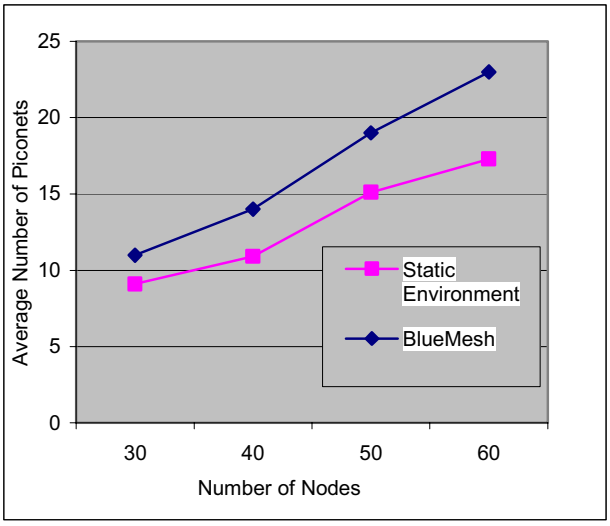

Figure 11. Number of Piconets comparison with BlueMesh.

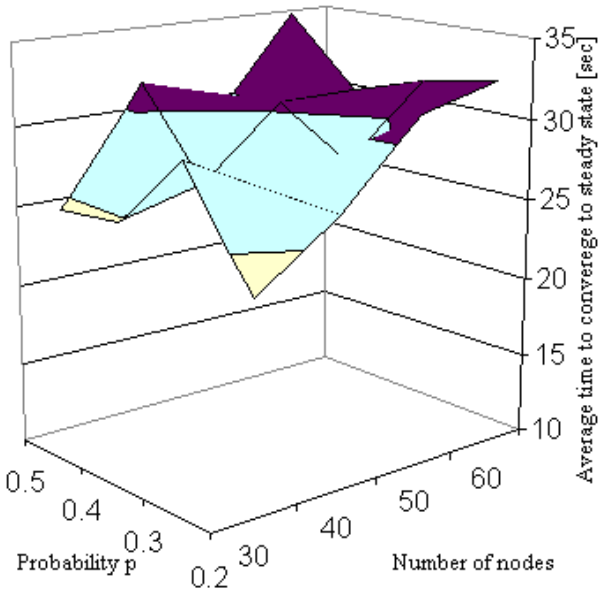

Figure 12. Average time to converge to a steady state.