BTSpin - Single Phase Distributed Bluetooth Scatternet Formation

Joy Ghosh, Vivek Kumar, Xin Wang, Chunming Qiao
Department of Computer Science and Engineering
University at Buffalo, The State University of New York
201 Bell Hall, Buffalo, NY 14260-2000
Email:{joyghosh, vkumar2, xwang8, qiao}@cse.buffalo.edu

***Abstract*— The Bluetooth standard specifies the formation of Piconets but only alludes to the possibility of joining several of these Piconets to form a Scatternet. In an attempt to formalize this Scatternet formation process, several schemes have been suggested. The centralized schemes suggested so far, require all the nodes to be in the radio range of each other (i.e. *single hop*) to ensure the correctness of their algorithm. To address multi hop scenarios, the distributed schemes suggested so far, usually require multiple phases to form a Scatternet. In particular, they need a considerable amount of time and energy in the topology discovery phase, during which the nodes exchange one hop or even two hop neighbor information. This also restricts these schemes' ability to efficiently cope with fully dynamic topology changes due to nodes joining/leaving. In this work, we propose a novel and practical approach called BlueToothSpin (BTSpin) to overcome several of the above mentioned shortcomings. BTSpin has a single phase, wherein the nodes concurrently form Scatternets and route data traffic. Our simulations show that BTSpin has a low Scatternet formation delay, and can form efficient multi-hop Scatternets when nodes arrive both *Incrementally* and *En Masse* (all at the same time). The total number of Piconets formed and the average number of roles per node (number of Piconets a node participates in) are also shown to be lower than other proposed mesh based protocols.**

I. Introduction

Bluetooth ([1], [2]) is a wireless communication system used to form personal area networks over a short range. On being turned on, a Bluetooth device can be in any one of the following modes: Inquiry, Inquiry Scan, Symmetric Inquiry. In the Symmetric Inquiry mode, a Bluetooth device toggles between the Inquiry mode and the Inquiry Scan mode. A device in the Inquiry mode sends out ID packets in a predetermined frequency hopping sequence. A device in the Inquiry Scan mode, while scanning frequencies in a preset frequency hopping sequence, waits for those ID packets. To ensure a frequency match between devices in these two modes, the device in the Inquiry mode hops frequencies twice as fast as the one in the Inquiry Scan mode. On a frequency match, the device in the Inquiry Scan mode responds with its Frequency-Hop Synchronization (FHS) packet containing its device ID and Bluetooth clock, and enters a Page Scan mode. The inquiring device, on receiving this FHS packet, pages the inquired device with its own clock. On receiving the page, the inquired device tunes its clock to that of the paging device, thereby becoming its Slave. This way, a star shaped network called Piconet may be formed with one Master and at most seven active Slaves. In the event of having more than seven Slaves, a Master can park/un-park Slaves so that only seven remain active at any given time. All intra-Piconet data transfer is coordinated by the Master, who periodically polls all its active Slaves as per as its scheduling policy. Slaves can only communicate with their Master. Only Masters can initiate this communication by the polling mechanism. Hence all traffic flows through the Master. To accommodate a greater number of active devices, the Bluetooth standard alludes to the possibility of joining several Piconets to form a larger network called a Scatternet. To this end, a node can have multiple roles in multiple Piconets to serve as a Bridge in between them. For example, it can be a Master in one of the Piconets and a Slave in all the other Piconets. Details of achieving such a bridged network (Scatternet) however, is not mentioned in the specifications. In an attempt to formalize this Scatternet formation process, many schemes have been described in the literature.

The solutions proposed so far can be broadly classified into two categories: *Centralized* and *Distributed*. Centralized schemes (e.g., [3], [4], [5], [6]) assume the entire network to be in the radio range of each other (single hop network). Thus these protocols can leverage on the complete network topology information to optimize the Scatternets formed. However, given that the average transmission radius of a Bluetooth device is only 10 meters, such a single-hop assumption may be too restrictive. To overcome this problem, distributed schemes (e.g., [4], [7], [8]) were proposed, which form multi-hop Scatternets. In most of these schemes, however, the nodes form a Scatternet in multiple phases and are unable to account for nodes that turn on (join) and turn off (leave) at different times. Moreover, any protocol requiring multiple steps or phases to form a Scatternet faces the challenge of selecting optimal phase timeout values (see III-A for more details).

In this work, we propose a novel framework called BlueToothSpin (BTSpin), which is a single phase distributed Scatternet formation strategy in Bluetooth. BTSpin makes use of a greedy and aggressive approach to form and heal mesh-based Scatternets, and allows concurrent data communication between the nodes. It does not require all nodes to be in communication range with each other, and as such, supports multi-hop networks. The Spin technique that we propose allows a Piconet to connect omni-directionally. It also promotes

fairness by relieving any particular node from being burdened by the Scatternet formation task more than the others in its own Piconet. At the same time, the absence of multiple phases in the protocol makes distributed implementation easier (by not having to worry about synchronizing phase timeout values). BTSpin tries to minimize the number of Piconets formed to result in a minimum number of Bridge nodes, as well as reduce the energy consumed in the process of forming (and breaking) Piconets. In Bluetooth, two nodes cannot exchange information without the existence of a Piconet between them. This constraint mandates that every new node, which attempts to communicate with another node, needs to form a Piconet with the other node. The process of Piconet formation requires a frequency match in the time domain between the two devices. As described earlier, both devices hop in a predetermined frequency hopping sequence, which implies that both devices must spend a considerable time in this process to ensure a good probability of a match. Often such Piconets are broken as soon as the information exchange is over. Thus, these Piconets do not form a part of the final Scatternet. Such Piconets shall be henceforth referred to as *temporary* Piconets in this paper. Since Bluetooth devices are particularly power constrained, BTSpin tries to eliminate the need to form *temporary* Piconets at any point in an effort to conserve energy. BTSpin also uses the concept of Backup Gateways that help in recovering from any single node failure (e.g., caused by turning off a node). To the best of our knowledge, no other solution has been proposed so far that is as capable as BTSpin in forming efficient multi-hop Scatternets for Bluetooth devices.

The rest of the paper is outlined as follows. In Section II, we present a thorough survey and discussion on related work. In Section III, we describe the details of the BTSpin protocol. In Section IV, we analyze and compare BTSpin with other protocols and present our simulation results. In Section V, we mention our future direction and conclude this work.

II. Related Work

Recently, the Bluetooth Scatternet formation problem has attracted a lot of attention. Broadly, Scatternet formation strategies can be classified into the following categories:

- *Centralized*: A special node (e.g. co-coordinator, leader) initiates and oversees the Scatternet formation process
- *Distributed*: Each node independently participates in the formation of a Scatternet, based only upon its local knowledge about the network
- *Single-Hop*: All nodes are in the radio range of each other
- *Multi-Hop*: Only some nodes are in the radio range of each other
- *Static*: All nodes arrive (or are turned on) simultaneously (or within a short time interval), and no nodes will leave (or are turned off)
- *Dynamic*: Nodes can leave/join the network at different times

Based on the above classification, the Scatternet formation schemes in the current literature can be summarized by the following Table I.

TABLE I
Classification of Scatternet formation protocols

	Centralized	Distributed
Single-Hop Static	[3], [9] [11], [5], [7]	[7], [10] [12]
Single-Hop Dynamic	None	[4], [13], [14] [6], [15], [16]
Multi-Hop Static	[11]	[8], [17]
Multi-Hop Dynamic	None	[18][1] BTSpin[2]

Centralized schemes permit the application of traditional graph theoretic techniques to optimize the generated Scatternet topology for data traffic. They assume the presence of a special node to gather the topology information, construct the topology graph and start the Scatternet formation process. Typically such special nodes are chosen by a leader election process. This process relies on empirical timeout values to ensure the correctness of the algorithm, and is difficult to auto-tune in real ad hoc scenarios. Further, gathering topology information requires the formation of a large number of *temporary* Piconets, which implies a large Scatternet formation delay. Once the initial topology has been determined, it is impractical to re-run the entire topology discovery phase to account for node leaving and joining the network at different times.

Salonidis *et al.*([3], [9]) were among the first who worked on this subject. They presented a centralized approach, assuming all devices to be turned on within a short time window. Marsan *et al.*[11] presented an offline topology construction scheme by formulating it as a min-max optimization problem. Stating it to be an NP Complete problem they presented a distributed approach with an interesting application of unused bits in the FHS packet. In [5], the authors assumed a single-hop scenario and approached the problem of Scatternet topology formation through graph theoretic formulation. They concentrated on graphs that can be partitioned into disjoint sets of edges, each of which is a (near-) perfect matching of the original graph. However, their approach has a multi-phased implementation similar to the one described in [3] and suffered from all the consequent disadvantages, including the problem of requiring appropriate empirical timeout values.

To mitigate these disadvantages of centralized approaches, and to deal with dynamic nature of networks (with respect to nodes joining and leaving the network at different times), several distributed approaches have been proposed in the literature. In contrast to the centralized schemes, current distributed approaches rely on local optimizations through information exchange with neighbors. This leads to wastage of resources due to large number of *temporary* Piconets. Also, distributed approaches, relying only on local information about

[1]partially dynamic: allows nodes to join later; cannot handle nodes leaving
[2]fully dynamic: allows nodes to join/leave later; our proposed protocol

the network, often end up in forming disconnected topologies, with several components in the topological graph. Hence most of these protocols cannot generate a single Scatternet even in the presence of physical connectivity.

Among the distributed approaches, Foo *et al.*([10], [12]) proposed a ring based topology. They leveraged the ease in routing and 'two-connectivity' (at least two nodes need to fail to cause the network to partition) to improve performance. Their protocol forces each node to be associated with two Piconets, thereby incurring expensive scheduling penalties. Petrioli *et al.*[8] proposed a multi-hop and distributed mesh topology construction. Their approach suffers from an expensive topology discovery phase and is intrinsically dependent on empirical timeouts which can be difficult to auto-tune in an ad hoc scenarios. To their credit, they generated a single Scatternet whenever physical connectivity was available. Each Piconet however, could end up having more than seven Slaves. This problem has been solved in [19], where the author addressed the problem of reducing the total number of Slaves in a Piconet using a distributed degree reduction technique. This approach requires each device to know its geographic location and thus mandates dependency on additional hardware viz. GPS [20] receiver. Yun *et al.*[18] proposed a star shaped topology construction scheme. Their approach required all the nodes in a Scatternet to be in Inquiry Scan, while the 'free' nodes could engage themselves in both Inquiry and Inquiry Scan. The links were formed based on memberships to Piconet groups. Only one bridge to another Piconet group was allowed. Since Inquiry Scan is a resource intensive task, forcing all the nodes in a Scatternet to be in Inquiry Scan waste resources. It also had an initial neighbor discovery phase that introduced the problem of requiring empirical timeout values as discussed earlier for the centralized schemes. Zaruba *et al.*[7] had proposed a Scatternet formation heuristic to induce a tree topology. It was a multi-phased approach with a time consuming election phase. Tan *et al.* ([6], [15], [16]) had a similar distributed tree formation protocol, where only certain types of nodes were allowed to connect with each other. They addressed the problem of nodes leaving/dying by allowing disconnected components to merge. They suffered from *En Masse* arrival problem (nodes arriving all at once) which in their particular case results in many disconnected partitions. Basagni *et al.*[17] proposed a multi-hop three-phased protocol similar to the one in [8], where they restrict the topology discovery phase to single hop neighbors. They relied on device ID and a distributed election process (as in [3]) to determine the roles (Master, Slave, Bridge) of nodes. Since Bluetooth requires the existence of a Piconet for any information exchange, their algorithm resulted in a large number of *temporary* Piconets. Having multiple phases, their scheme had to rely on empirical timeout values for protocol correctness. Also having a separate topology discovery phase, they too failed to account for nodes joining and leaving the network.

Kapoor *et al.*[21] approached the problem from a network capacity point of view and provided insights into the performance of different topologies. They however, did not discuss techniques to form any of those topologies. Liu *et al.*([14], [13], [4]) proposed a randomized distributed strategy for forming a chain like Scatternet. Their algorithm relied on all nodes being in the radio range of each other. They addressed the problem of incremental joining of nodes, but failed to account for nodes leaving (turning off) the network at different times.

It is evident from the discussion above that even the existing distributed techniques like [4], [10], [7] can only be used for single hop networks. The schemes in [8], [18], [17] on the other hand do address multi-hop scenarios but suffer from an resource intensive multi-phased approach that fails to account for dynamic node joining/leaving.

III. BTSpin Strategy

The main goal of BTSpin is to provide an efficient solution in a realistic ad hoc scenario. Hence we adopted the Mesh-based Scatternet topology as it has been proved to be superior to Tree-based Scatternet formation protocols ([17], [8]). BTSpin employs the proposed Spin concept (see III-B below), which allows data communication and Scatternet formation to be performed concurrently. This strategy avoids un-necessary *temporary* Piconets that are formed for the sake of mutual information exchange. In case a Piconet formed is redundant and needs to be broken, we keep the Backup Gateway information to help maintain a single Scatternet even after a single point Bridge node failure. Since the nodes do not have to go through an initial phase where they discover topology by exchanging one hop or even two hop neighbor information, the Scatternet formation delay is low and Piconets can actually start exchanging data earlier. We shall now proceed to explain in detail the working of this BTSpin protocol.

A. Single-Phase vs. Multi-Phase Scatternet Formation

In most mesh-based Scatternet formation techniques described in literature, the nodes have multiple phases to form a Scatternet. For example, in [17], the authors proposed a three phased approach. The first phase does topology discovery, wherein all nodes exchange information with their physical neighbors (i.e., in the radio range). As discussed earlier, this requires a large number of *temporary* Piconets, making this phase resource intensive in terms of energy consumption. In the second phase, a distributed Piconet formation algorithm is used. This phase additionally requires each node to updates all its physical neighbors about its assigned role (Master or Slave) and its Piconet ID (the node ID of the Master of the Piconet). Once the Piconets are formed, the Masters use their Slave's one-hop neighbor information to decide network Bridges that join Piconets into a single Scatternet in the final phase. This protocol suffers from the fact that a Piconet could end up with more than seven Slaves. In [8], the authors proposed a similar approach, but with only two phases. The topology discovery done in the the first phase is more extensive than the one described in [17]. More specifically, all the nodes first discover their one-hop neighbors and then exchange that

information once again with all one-hop neighbors to get two-hop neighbor information. Thus, it ends up forming a larger number of *temporary* Piconets than in [17]. In the second phase, the Scatternet is formed. It is evident that the phased approach requires the use of empirical timeout values for each phase. These timeout values are intrinsically dependent on the node density, and are therefore difficult to auto-tune in a real ad hoc scenario. Thus during any topology discovery phase, if a node times out early, it may not discover all its neighbors. Alternatively, if the timeout value is large, the Scatternet formation delay is increased. Moreover, once this topology discovery phase is concluded, the protocol fails to accommodate any node that joins (or is turned on) the network at a later time. In contrast to these approaches, BTSpin has a single Spin phase (see III-B below) and forms a multi-hop Scatternet through a distributed mechanism, which does not require a complete knowledge of one-hop or two-hop neighborhood information. BTSpin can also account for nodes joining (turning on) and leaving (turning off) the network at different times.

B. Spin Technique

The main feature of the BTSpin strategy is the proposed Spin technique that each Master in BTSpin employs to achieve a balance between Scatternet formation and intra-Piconet data communication. We define a single 'Spin' to comprise of an Inquiry mode followed by an Inquiry Scan mode. The Master in each Piconet schedules each of its Slaves for a Spin in a round robin fashion. While a Slave is spinning, the Master can continue data communication with the other Slaves. The spinning node also obtains relevant Piconet parameters from its Master (e.g., size of Piconet, Master ID) before it starts its spin. As detailed in the algorithm below (see III-C), on discovering another spinning node, this information is used to make intelligent decisions on when to switch roles, form a new Piconet, etc. The actual implementation of this Spin can be achieved by requiring the Master to put the Slave in 'hold' mode (as in the Bluetooth Specification) for a Spin duration. In a 'hold' mode, the Slave gives up its active Piconet membership and listens to its Master only at predetermined points in time. Once the Slave completes its spin, it notifies the Master. The Master then selects yet another Slave to start its Spin. The Master himself goes for a Spin once every Slave has had its turn, and then the entire process is repeated. In this way, the Piconet has a chance to connect omni-directionally with other Piconets and Free nodes, without burdening any single node in the Piconet with the task of trying to connect with other nodes in the network. Slave-Slave bridges are not scheduled by the Master to Spin since in BTSpin the maximum number of roles a node can have is restricted to two. This is done to improve system capacity (as analyzed in [21]). To keep the maximum number of Slaves in a Piconet to be no greater than seven, a Master removes itself from the spin schedule, once its Piconet gets full.

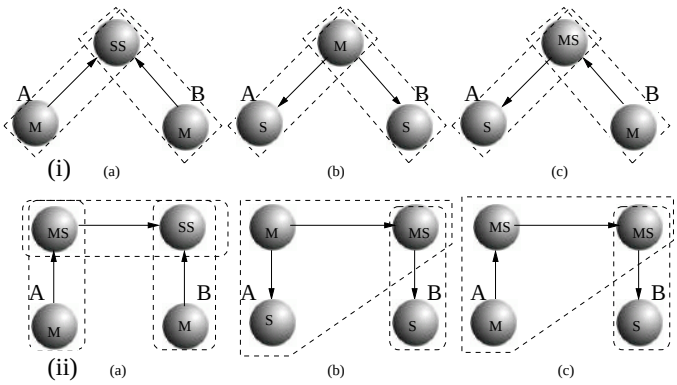

Fig. 1. (i) A and B are two-hop connected (ii) A and B are three-hop connected

C. Connect Rules and Role Determination

Initially all nodes are Free nodes when they turn on. All Free nodes continue to Spin till they connect with another node in the network, either as a Slave, or as a Master. Once a node becomes a part of a Piconet, it waits for the Master to schedule it to Spin (if it is a Slave), or it periodically Spins itself (if it is a Master). BTSpin lays out rules for two spinning nodes to determine their roles when they happen to connect for the first time (i.e. they are in the radio range of each other, with one of them in Inquiry mode and the other in Inquiry Scan mode). At first when two nodes connect, they form a Piconet between themselves, with the one in the Inquiry mode acting as the Master of the one in the Inquiry Scan mode. Further, depending on our connect rules, they determine their final roles and may do a role reversal (according to the Bluetooth specification) if required. Thus, the *initial* Piconet formed in between the two nodes that meet each other while spinning, may be broken, modified (by role reversals), or kept as it is first formed, depending on our role determination policy. For example, if two spinning Slave nodes meet each other (come in the radio range of each other and one happens to be in Inquiry mode, while the other is in Inquiry Scan mode), they form a Piconet between themselves (where one becomes a Master to the other). However, they might decide to break that Piconet and not remain connected with each other at all, if they realize their Masters to be already two-hop or three-hop connected (Fig. 1). Such scenarios lead to formation of *temporary* Piconets (i.e., Piconets that need to be broken soon after they are initially formed, and are not part of the final Scatternet) in the process of Scatternet construction. In this paper we shall henceforth refer to the following definitions:

- ***connected*** nodes: nodes that are either directly, two-hop or three-hop connected to each other
- ***atomic*** Piconet: that contains a Master with a single Slave

Following, we will describe our algorithms for each of the node types that are allowed to spin (i.e., Free, Slave, Master, and Master-Slave Bridge). Rules have been laid out for them to determine their roles (i.e., Slave, Master, or a Bridge), upon forming an *initial* Piconet with (or connecting with)

another spinning node, as mentioned above. We also discuss the conditions that require the *initial* Piconet to be broken.

*1. If the spinning node is **Free***

a) On connecting with another spinning **Free** node:

Both nodes initially form a Piconet, with the node in the Inquiry mode acting as the Master of the other node in the Inquiry Scan mode. This Piconet remains and both Master and Slave take turns to Spin and also communicate with each other.

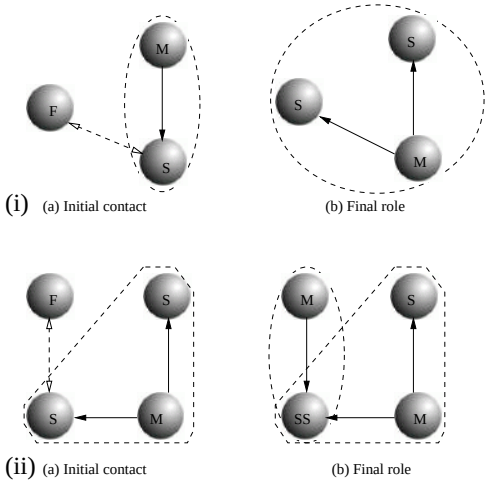

Fig. 2. (i) Free node becomes Slave (ii) Free node becomes master

b) On connecting with a spinning **Slave** node:

If the peer Slave formed an ***atomic*** Piconet with its own Master, the peer performs a role reversal (as described in the Bluetooth Specification) to become a Master of both the Free node (which turns into a Slave) and its original Master, to form one single Piconet (Fig. 2(i)). Otherwise, the peer Slave becomes a Slave-Slave Bridge node (Fig. 2(ii)) of the Free node, which then becomes a Master. The Free node is not made a Slave of its peer in the second case. This is done to reduce the number of Piconets that would be formed if the peer Slave turned into a Master-Slave Bridge, which could potentially spin. Note that to restrict the maximum number of roles of any node to two, Slave-Slave Bridges in BTSpin are not scheduled to Spin. This is due to system capacity considerations, as analyzed in [21].

c) On connecting with a spinning **Master** node:

The Free node becomes a Slave of the peer.

d) On connecting with a spinning **Master-Slave** node:

The Free node becomes a Slave of the peer.

The role determination process for a Free node can be presented as follows.

FreeNodeRoleDetermination()

```
1   if (peerRole = Free)
2       role[node in Inquiry] = Master;
3       role[node in Inquiry Scan] = Slave;
4   else if (peerRole = Slave)
5       if (peer is only Slave of its Master)
6           peerRole = Master; (role reversal)
7           myRole = Slave; (Fig. 2(i))
8       else
9           peerRole = Slave-Slave Bridge;
10          myRole = Master; (Fig. 2(ii))
11      end if
12  else if (peerRole = Master OR Master-Slave Bridge)
13      peerRole = Same as before;
14      myRole = Slave;
15  end if
```

*2. If the spinning node is a **Slave***

a) On connecting with a spinning **Free** node:

This scenario has already been described in the above routine for a Free node.

b) On connecting with another spinning **Slave** node:

If the Masters of these two Slaves are ***connected*** (directly, two-hop or three-hop) with each other, the initial Piconet formed between these two Slaves will be broken, to minimize the number of Bridge nodes. Otherwise, the Slave belonging to a larger Piconet becomes a Slave (Slave-Slave Bridge) of its peer, which then becomes its peer's Master (Master-Slave Bridge) ((Fig. 3(i)). The motivation behind this role determination is the fact that a Master-Slave Bridge being a Master of a Piconet has more responsibilities (e.g., scheduling Slaves for spinning, polling) than a Slave-Slave Bridge. Hence the Slave belonging to a smaller Piconet is a better fit to this additional task. The Slave turning into a Master-Slave Bridge also checks to see if it formed an ***atomic*** Piconet with its own Master, in which case it performs a role reversal to become a Master of both its peer and its own Master (Fig. 3(ii)). The role reversal helps reduce the number of Piconets that need to be formed.

c) On connecting with a spinning **Master** node:

If this Slave finds its own Master to be a Slave of the peer, it leaves its old Master and becomes a Slave of the peer (Fig. 4(i)). On the other hand, if this Slave finds its Master to be two-hop or three-hop connected to the peer, the initial connection is broken, to avoid short loops in the mesh. Otherwise, this Slave becomes a Slave (Slave-Slave Bridge) of its peer (Fig. 4(ii)).

d) On connecting with a spinning **Master-Slave** node:

This scenario is identical to the one described above.

The role determination process for a Slave node can be presented as follows.

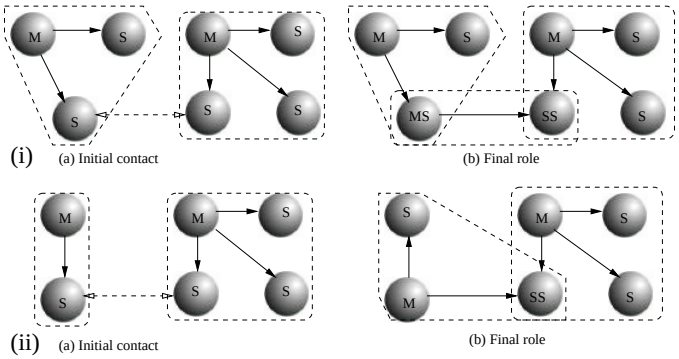

Fig. 3. (i) Slaves make new Piconet (ii) Old Piconet is re-arranged

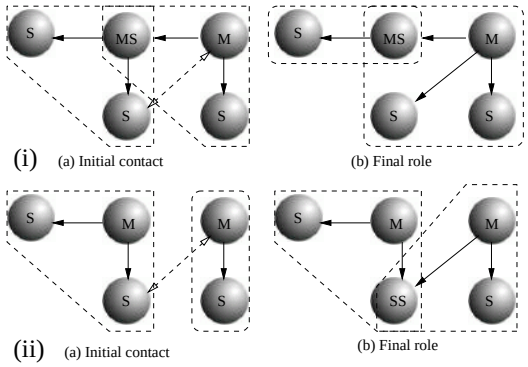

Fig. 4. (i) Slave joins new Piconet leaving old one (ii) Slave becomes a bridge

SlaveNodeRoleDetermination()

```
1  if (peerRole = Free)
2      As described in FreeNodeRoleDetermination()
3      when Free node meets Slave node
4  else if (peerRole = Slave)
5      if (my Master is connected to peer's Master)
6          break this current Piconet
7          do not connect
8      else
9          role[Slave of bigger Piconet] = Slave-Slave Bridge;
10         role[other Slave] = Master-Slave Bridge; (Fig. 3(i))
11         if (new Master-Slave Bridge is an only Slave)
12             Master-Slave Bridge does role reversal to be
13             Master of peer and its own Master (Fig. 3(ii))
14         end if
15     end if
16 else if (peerRole = Master OR Master-Slave Bridge)
17     if (my Master is a Slave of peer)
18         leave my old Master; make peer my new Master;
19         roles remain same for both peer and me; (Fig. 4(i))
20     else if (my Master is connected to peer)
21         break this current Piconet
22         do not connect
23     else
24         peerRole = Same as before;
25         myRole = Slave-Slave Bridge; (Fig. 4(ii))
26     end if
27 end if
```

*3. If this spinning node is a **Master***

a) On connecting with a spinning **Free** node:

This scenario has already been described in the above routine for a Free node.

b) On connecting with a spinning **Slave** node:

This scenario has already been described in the above routine for a Slave node.

c) On connecting with another spinning **Master** node:

If these Masters are already two-hop connected with each other, they retain this direct connection if and only if one of them has the connecting bridge node as its only Slave. In that case, that Master relieves its Slave and becomes a Slave of its peer, thereby merging the two Piconets (Fig. 5(i)). If on the other hand, the two Masters are not two-hop but three-hop connected with each other, they may still retain this direct connection, if and only if the pair of Slaves that form the two bridges connecting them, have an ***atomic*** Piconet between themselves. In that case, these two Masters instruct their corresponding Slaves to break that ***atomic*** Piconet. The Master of the smaller Piconet becomes the Slave (Master-Slave Bridge) of the Master of the bigger Piconet (Fig. 5(ii)). If these two Masters are not ***connected*** at all, they remain connected directly with each other with roles as mentioned above.

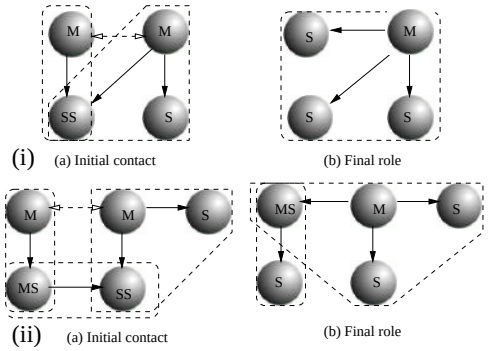

Fig. 5. (i) Piconets get merged (ii) Intermediate Piconet removed (ii) Smaller master becomes bridge

d) On connecting with a **Master-Slave** node:

This scenario is almost identical to that of a Master connecting with another Master, except for a few additional checks. If the Master node finds the Master-Slave Bridge to be its own Slave, the initial Piconet formed between these two nodes is broken. Else, the Master retains direct connection with the peer almost as it would with another Master (Fig. 6(i)). If the peer's Master is a Slave of this

Master, the peer leaves its old Master and joins this Master (Fig. 6(ii)). If the Master eventually retains this direct connection with the Master-Slave Bridge node, it becomes a Slave (Master-Slave Bridge) of the peer irrespective of their Piconet sizes (Fig. 6(iii)). This is because we try to restrict the number of roles that a node can have to at most two to improve system capacity (as analyzed in [21]).

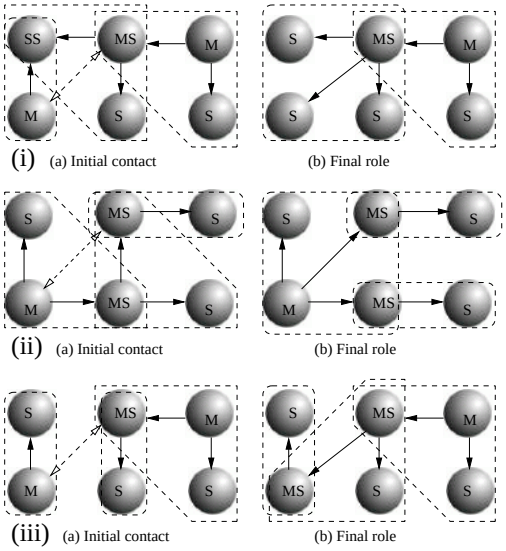

Fig. 6. (i) Piconets get merged (ii) Piconet re-arranged (iii) Master becomes bridge

The role determination process for a Master node can be presented as follows.

MasterNodeRoleDetermination()

```
1   if (peerRole = Free)
2       As described in FreeNodeRoleDetermination()
3       when Free node meets Master node
4   else if (peerRole = Slave)
5       As described in SlaveNodeRoleDetermination()
6       when Slave node meets Master node
7   else if (peerRole = Master)
8       if (I am two-hop connected to peer)
9           if (me OR peer has only one Slave)
10              Master with one Slave releases Slave;
11              role[Master with one Slave] = Slave;
12              role[other Master] = Master; (Fig. 5(i))
13          else
14              break this current Piconet
15              do not connect
16          end if
17      else if (I am three-hop connected to peer)
18          if (our Slaves form an atomic Piconet)
19              instruct Slaves to break their atomic Piconet
20              role[smaller Master] = Master-Slave Bridge;
21              role[bigger Master] = Master; (Fig. 5(ii))
22          else
23              break this current Piconet
24              do not connect
25          end if
26      else
27          role[smaller Master] = Master-Slave Bridge;
28          role[bigger Master] = Master;
29      end if
30  else if (peerRole = Master-Slave Bridge)
31      if (peer is my Slave)
32          break this current Piconet
33          do not connect
34      else
35          Similar to when Master meets Master; (Fig. 6(i))
36          Except for a couple of checks and differences
37          if (peer's Master is my Slave)
38              peer leaves old Piconet and joins mine;
39              our roles remain the same; (Fig. 6(ii))
40          end if
41          if (they DO connect)
42              peerRole = Same as before;
43              myRole = Master-Slave Bridge;
44              irrespective of our Piconet sizes; (Fig. 6(iii))
45          end if
46      end if
47  end if
```

*4. If this spinning node is a **Master-Slave Bridge***

a) On connecting with a spinning **Free** node:
This scenario has already been described in the above routine for a Free node.

b) On connecting with a spinning **Slave** node:
This scenario has already been described in the above routine for a Slave node.

c) On connecting with a spinning **Master** node:
This scenario has already been described in the above routine for a Master node.

d) On connecting with another spinning **Master-Slave** node:
By definition, a Master-Slave Bridge belongs to two Piconets. Since in BTSpin we restrict the number of roles of a node to two, the initial Piconet formed between the two nodes shall be broken.

The role determination process for a Master-Slave Bridge node can be presented as follows.

Master-SlaveBridgeNodeRoleDetermination()

```
1   if (peerRole = Free)
2       As described in FreeNodeRoleDetermination()
3       when Free node meets Master-Slave Bridge node
4   else if (peerRole = Slave)
5       As described in SlaveNodeRoleDetermination()
6       when Slave node meets Master-Slave Bridge node
```

```
7   else if (peerRole = Master)
8       As described in MasterNodeRoleDetermination()
9       when Master node meets Master-Slave Bridge node
10  else if (peerRole = Master-Slave Bridge)
11      Master-Slave Bridge nodes already have two roles
12      Hence break this current Piconet
13      do not connect
14  end if
```

D. Backup Gateways

In the multi-phased mesh-based approaches proposed in literature ([17], [8]), we observed a large number of *temporary* Piconets being formed between nodes, just for exchanging symmetric information about their one-hop or two-hop neighbors. BTSpin minimizes the number of such *temporary* Piconets that need to be formed. In most of our cases, when two spinning nodes connect with each other for the first time (i.e. come in the radio range of each other with one node in Inquiry mode and the other in Inquiry Scan mode), they form and keep the Piconet, either as it was initially formed, or modifying it (if mandated by our role determination policy) by performing a role reversal. Only in a few cases, determined by our connect rules and role determination policy, two nodes might need to break the Piconet that is formed initially. Even in those circumstances, BTSpin tries to salvage as much as possible by keeping a Backup Gateway information for later use. When two (bridge) nodes break the link between them, the Masters of their respective Piconets (if they belong to one) keep a note of these two nodes as Backup Gateway bridges. In the future, if these two Masters ever need to connect with each other (due to their existing Bridge/Bridges failing or turning off) they can make use of this Backup information to know precisely which Slave/Slaves to instruct to act as Bridge/Bridges between these two Piconets. This is the adaptive nature of BTSpin strategy that takes care of nodes leaving (turning off) the network.

E. Node Failures

As we mentioned in the section above, BTSpin can adapt itself to heal the Scatternet when nodes leave (fail or turn off) the network. Following is the action taken on failures of different types of nodes:

1. Slave node

The Master of this Slave eventually considers the link as dead when the Slave misses successive polls. No further action is necessary.

2. Master node

The Slaves realize that the Master is down if they are not polled successively for some predetermined period of time. The nodes that were not Bridges act as Free nodes and start spinning till they connect with someone else. The Bridge nodes continue with their their second role in the other Piconet.

3. Bridge node

If a Slave-Slave Bridge goes down, both Masters use the Backup Gateway information to find another Bridge node if one is available. If a Master-Slave Bridge goes down, the Master of the Bridge notes the absence as it would for any of its other Slaves. The Slaves of the Bridge behave as described above for the failure of a Master node.

F. Routing over BTSpin

Since BTSpin generates a mesh based multi-path multi-hop Scatternet, any flooding based routing scheme can be used. The multi-path topology provides the advantage of fault tolerance and possibility of data stripping (data aggregation). To overcome loops in the mesh, data sequencing is necessary to identify duplicate data packets.

IV. Simulation Results

To evaluate the efficiency of the proposed protocol, we implemented BTSpin in GloMoSim [22]. In Bluetooth, the dynamic nature of the network is driven more by turning on and off the devices at different times than by mobility of the devices. Hence we simulated a static network (i.e., no node mobility), but considered node joining/leaving (as a result of turning on/off nodes) in three different physical network topologies.

A. En Masse vs. Incremental Arrival

The three topologies with varying degrees of randomness considered in our simulations (see detailed description below) are:

- GRID: No randomness in node placement. Nodes are placed uniformly in a grid.
- CELL: Some randomness in node placement. Nodes are placed at random in cells.
- RANDOM: Complete randomness in node placement. Nodes can be placed anywhere in the terrain.

In each of the above scenarios, we simulated BTSpin with both 'Incremental' arrival (joining) of nodes into the network and *En Masse* arrival. The terrain was taken to be a square region with varying sizes with each node having a transmission range of 10 meters. The simulation was run multiple times with varying seeds, each for a duration of 300 seconds.

1. GRID

In this scenario, 36 nodes were placed uniformly onto a grid that covered the entire terrain. The grid unit size was chosen to be 7 meters. Since the transmission range of each node was fixed to 10 meters, the nodes could only contact their (maximum of 8) immediate grid neighbors.

Fig. 7 shows the Scatternet formed when the nodes arrive (i.e., devices get turned on) all at the same time (*En Masse*), whereas Fig. 8 shows the Scatternet formed when nodes join the network at the rate of one node every 10 seconds (*Incremental* arrival) in the order of their node ID shown in the figure. We find that in the case of incremental arrival, the number of Piconets formed (14 in Fig. 8) were smaller

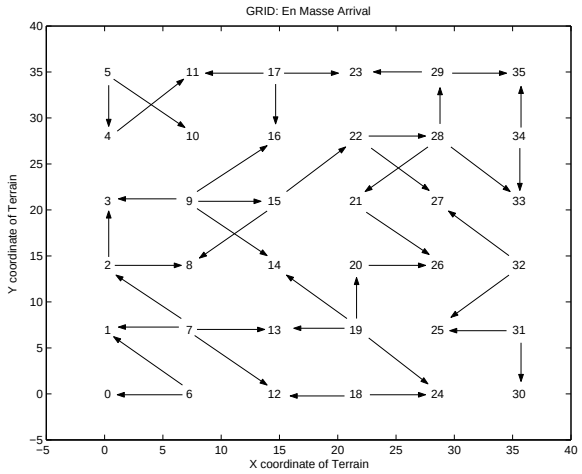

Fig. 7. En Masse arrival of nodes in the GRID physical network topology

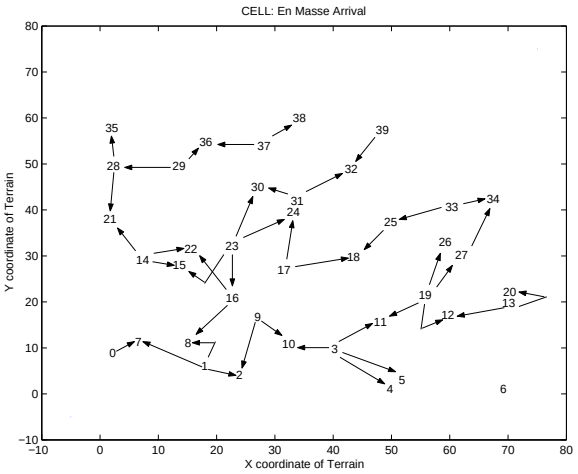

Fig. 9. En Masse arrival of nodes in the CELL physical network topology

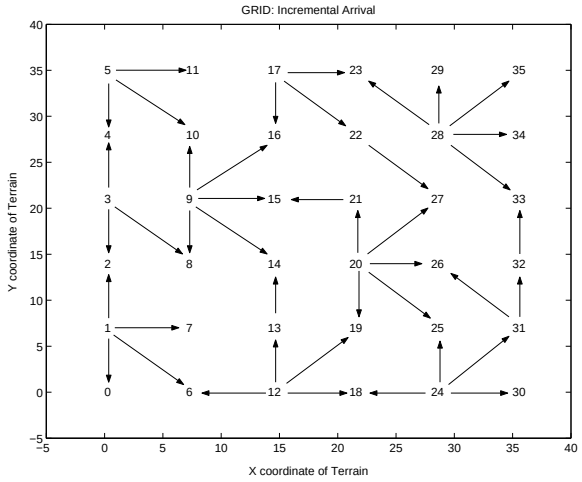

Fig. 8. Incremental arrival of nodes in the GRID physical network topology

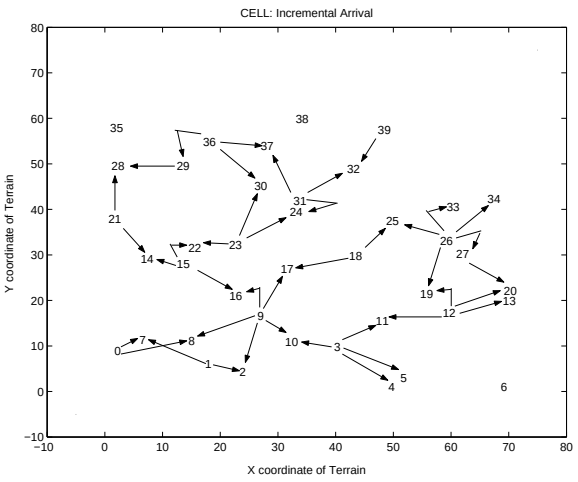

Fig. 10. Incremental arrival of nodes in the CELL physical network topology

with Piconets having a larger number of slaves in a Piconet on average. This is due to the fact that in the case of *En Masse* arrival, Piconets are formed in parallel at multiple locations and they may not merge with each other. Hence, the number of Piconets formed (18 in Fig. 7) is larger, and to maintain connectivity, the Scatternet has more bridge nodes, which also means a higher number of roles per node on average. The Scatternet formation delay in this *En Masse* case was observed to be 41.03 seconds (see more discussion in Section IV-B and IV-C).

2. CELL

Based on the number of nodes in the simulation, the physical terrain is divided into a number of cells. Within each cell, a node is placed randomly. In this case, we simulated with 40 nodes.

Fig. 9 illustrate the Scatternet formed when nodes arrive *En Masse*, and Fig. 10 illustrate the Scatternet formed when nodes join one after another in the order of their node ID (*Incremental* arrival). In both cases, we find the node 6 is left un-connected. This is because the node 6 is physically isolated from the rest (i.e. no other node was within a radius of 10 meters from it). In the incremental case we find node 35 is left un-connected too. This is because, at present BTSpin restricts each node to belong to at most two Piconets. This is done to increase the system capacity as analyzed in [21]. Hence in BTSpin, a Master never schedules a Slave-Slave Bridge to spin. Thus, node 35 cannot connect to its only neighbor (node 28) which already serves as a Slave-Slave Bridge. As before, the overall Scatternet formation with *Incremental* arrival has a lower overall number of Piconets and lower roles per node on average than the case with *En Masse* arrival. The Scatternet formation delay in the *En Masse* arrival case was observed to be 28.05 seconds. This is less than that in the GRID topology primarily due to the fact that the nodes in the CELL topology are less scattered as compared to those in GRID. Hence, in the CELL topology, a spinning node has on average a larger number of spinning

neighbors, thus increasing the chance of connecting with them. This reduces the time taken to form the Scatternet.

3. RANDOM

In this scenario we place the nodes completely at random anywhere within the terrain.We simulated 40 nodes for this scenario.

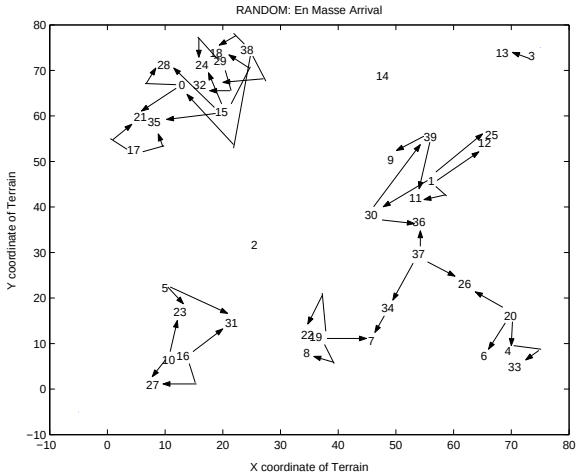

Fig. 11. En Masse arrival of nodes in the RANDOM physical network topology

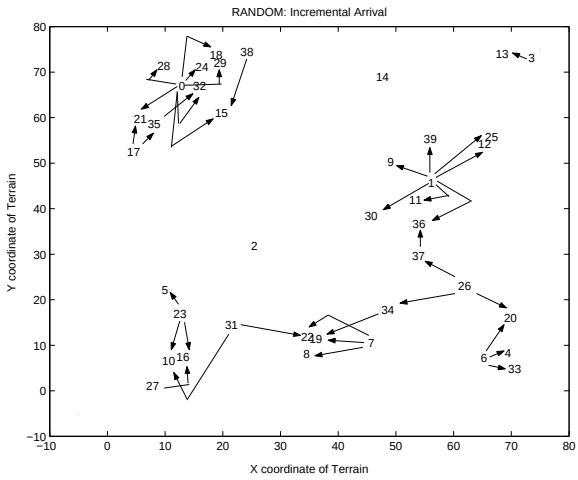

Fig. 12. Incremental arrival of nodes in the RANDOM physical network topology

Fig. 11 and Fig. 12 show the cases for *En Masse* arrival and *Incremental arrival* respectively. Note that whenever there is physical connectivity among nodes, BTSpin succeeds in forming a Scatternet. Given the node placement as shown in our simulation, we have a number of Scatternets formed (3 in Fig. 11 and 2 in Fig. 12). Nodes 2 and 14 remained unconnected since they were physically isolated from the rest of the nodes. As before, we find that in the incremental arrival the Piconets formed have more slaves on average than when all the nodes arrive at once.

B. Comparison with other single-hop approaches

Two of the important performance metrics are the initial connection setup delay (time taken by a Free node to join a Piconet for the first time) and the final Scatternet formation delay. We compared the delay in BTSpin with the following two schemes in [6]:

- PROB: A probabilistic scheme where each node becomes a Master with a probability of 0.5 and tries to connect with at most 5 Slaves.
- TSF: A Tree-based Scatternet formation protocol.

Both of these schemes are only suited for single-hop networks, where any node can potentially connect with any other node, anywhere within the network. This is not required by BTSpin. To evaluate the effect of this difference (single-hop vs. multi-hop) we compared these three schemes for the delays involved in Scatternet formation. In all the protocols, nodes are assumed to arrive all at once.

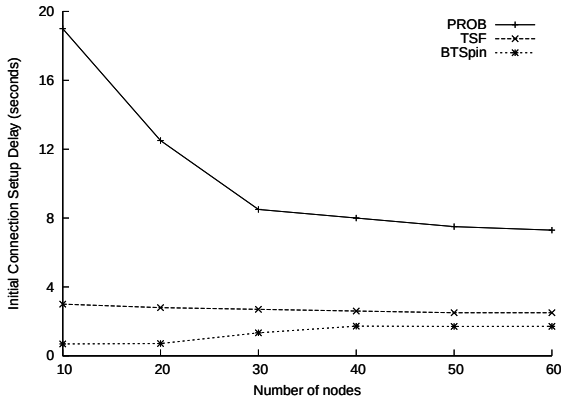

Fig. 13. Initial connection setup delay varying with number of nodes

Fig. 13 shows the initial connection setup delay for a Free node to join a Piconet for the first time. Both TSF and BTSpin outperform PROB. Since BTSpin employs a greedy approach in connecting the nodes together, the initial delay in BTSpin is less than that in TSF where Free nodes can only connect to other Free and non-Root nodes (refer to [6]). In BTSpin, the delay initially increased with the increase of the number of nodes due to our ‘Spin’ technique because of which the Free node had to wait for a neighbor node to be in the spin mode. With more than 40 nodes, that delay stabilizes due to the increased number of spinning nodes nearby.

Fig. 14 shows the final Scatternet formation delay. PROB has the lowest delay among the three, but due to its non-deterministic nature it cannot guarantee a fully connected Scatternet. The delay in BTSpin initially increases with the increase in the number of nodes, but with 40 nodes or more, it stabilizes. The overall delay in BTSpin remains much lower than that in TSF. In TSF, when the nodes arrive *En Masse*, several disconnected tree components are formed, which take a long time to merge with each other. This is because only the Root nodes can merge with other Root nodes (refer to [6] for further details). Thus its delay keeps increasing with an

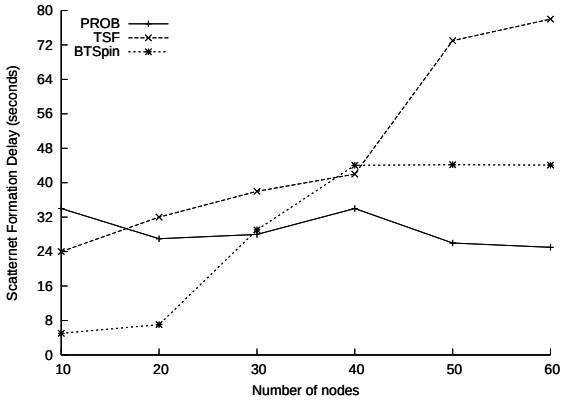

Fig. 14. Final Scatternet formation delay varying with number of nodes

increase in the number of nodes. In BTSpin with *En Masse* arrival, the number of Piconets formed increases with the number of nodes. Hence the delay for getting them connected increased initially. However, that delay stabilizes with a larger number of nodes due to increased node density.

C. Comparison with other multi-hop schemes

We compared BTSpin with two-phased mesh based protocol called BlueMesh ([8]). In both protocols, we considered the *En Masse* arrival. When analyzing the multi-hop Scatternets formed, we use the following two performance metrics:

- total number of Piconets formed
- average number of roles per node

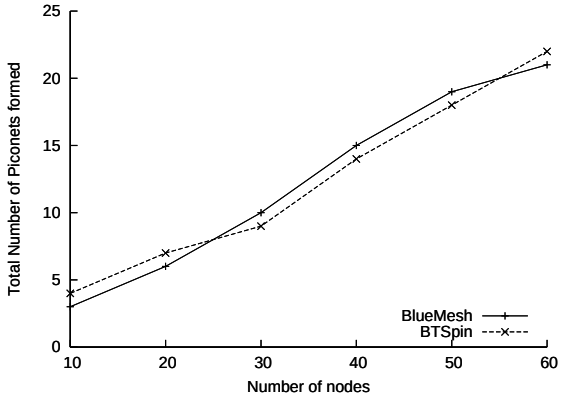

Fig. 15. Total number of Piconets varying with number of nodes

Among all the operations that a Bluetooth node performs, Piconet formation (Inquiry and Inquiry Scan) is the most resource intensive with respect to the time and energy spent by a node. Hence the number of Piconets formed during the process of Scatternet formation is of importance. Fig. 15 shows that BTSpin forms comparable number of total Piconets with respect to BlueMesh even though BTSpin does not engage in a multi-phase process including topology discovery which BlueMesh uses to reduce the number of Piconets contained in the Scatternet formed. However, the total number of Piconets formed in BlueMesh is higher than the number of piconets in BTSpin due to the large number of *temporary* Piconets that are formed during the topology discovery phase alone.

The number of Piconets formed in BTSpin is also close to the theoretical minimum. For example in one of our simulation runs , we had 40 nodes in a $50m * 50m$ terrain. Since the nodes were placed at random, the approximated average node density was:

$$\frac{40}{50 * 50} = 0.016 \ nodes/sq.m \tag{1}$$

nodes per square meter. With the transmission radius of 10 $meters$, we had approximately:

$$\lfloor \pi * 10 * 10 * 0.016 \rfloor = 5 \ nodes \tag{2}$$

in the radio range of each other. Thus on an average we could have one Master with four Slaves in a Piconet. As shown in [14], the lower bound on the number of Piconets in any Scatternet is:

$$\lceil \frac{n-1}{k} \rceil$$

where 'n' is the total number of nodes in the network and 'k' is the average number of Slaves in a Piconet. Thus in our case, the lower bound on the number of Piconets is:

$$\lceil \frac{40-1}{5-1} \rceil = 10 \tag{3}$$

while in the simulation we get 14 Piconets are formed in BTSpin.

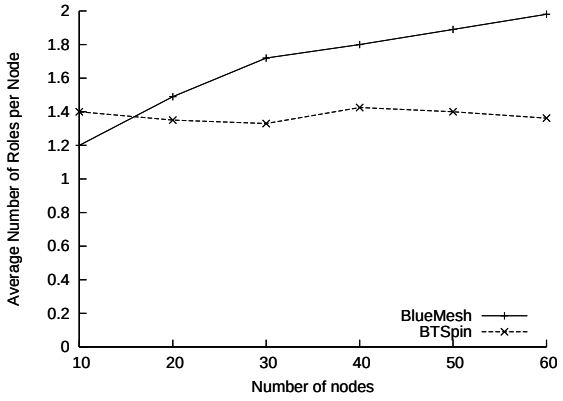

Fig. 16. Average number of roles per node varying with number of nodes

Fig. 16 shows the average number of Piconets that a node belongs to. If the average is lower, it implies a fewer number of Bridge nodes in the Scatternet. Since the Bridge nodes have to participate in multiple Piconets, they may become the bottleneck for data throughput. Hence, having fewer Bridges in a Scatternet is desirable, as long as all the Piconets can be connected into one single Scatternet. Clearly, BTSpin has fewer Bridges than BlueMesh when the number of nodes is reasonably large, but still ensures connectivity whenever the nodes are in each other's radio range.

V. Concluding Remarks

BTSpin is a single phase distributed Scatternet formation process that is effective in creating a mesh-based multi-hop Scatternet both with *En Masse* and *incremental*— arrival of the nodes. Unlike any of the existing protocols, it supports fully dynamic node joining/leaving without the need to rely on some central repository of the network topology information and without collecting one-hop or two-hop neighbor information through a resource consuming topology discovery. It employs the proposed Spin technique to achieve concurrency between Scatternet formation and data communication. The Scatternet formation delay is shown to be lower than some other existing schemes. The total number of Piconets formed and the average number of roles per node are shown to be smaller compared to other mesh based protocols, even though each Piconet does not contain more than seven Slaves. The Spin technique is fair to all the nodes in the Piconet and does not burden any particular node with the task of connecting with other nodes in the network. It also enables a Piconet to connect omni-directionally. BTSpin has an adaptive strategy to account for nodes leaving (turning off) the network at any time. It uses the proposed concept of Backup Gateways to quickly heal from a single node failure (caused by turning off a node). Thus, BTSpin lends itself as an attractive candidate for distributed and multi-hop Scatternet formation in Bluetooth.

Currently we are in the process of testing BTSpin's robustness against node failures by using the backup gateway information collected from all the failed node connections as described in Section. III-D. As an extension to the Scatternet formation process, we are also devising an efficient routing protocol that leverages the benefits of BTSpin technique.

References

[1] B. SIG, "The Bluetooth Radio System," *Specification of the Bluetooth System Version 1.2*, vol. I, II, November 2003.

[2] J. Haartsen, "The Bluetooth Radio System," *IEEE Personal Communications*, vol. 7(1), pp. 28–36, Feb 2000.

[3] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, "Distributed topology construction of Bluetooth personal area networks," *IEEE INFOCOM 2001, Anchorage, Alaska*, vol. 3, pp. 1577–1586, April 2001.

[4] C. Law, A. 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, Long Beach , CA*, pp. 183–192, October 2001.

[5] S. Baatz, C. Bieschke, M. Frank, C. Kuhl, P. Martini, and C. Scholz, "Building Efficient Bluetooth Scatternet Topologies from 1-Factors," *Proceedings of the IASTED International Conference on Wireless and Optical Communications, WOC 2002, Banff, Alberta, Canada*, pp. 300–305, July 2002.

[6] G. Tan, A. Miu, J. Guttag, and H. Balakrishnan, "Forming Scatternets from Bluetooth Personal Area Networks," *MIT Technical Report*, no. MIT-LCS-TR-826, October 2001.

[7] G. Zaruba, S. Basagni, and I. Chlamtac, "Bluetrees - Scatternet formation to enable Bluetooth-based ad hoc networks," *IEEE ICC 2001, Helsinki, Finland*, vol. 1, pp. 273–277, June 2001.

[8] C. Petrioli and S. Basagni, "Degree-Constrained Multihop Scatternet Formation for Bluetooth Networks," *Proceedings of IEEE Globecom, 2002, Taipei, Taiwan, R.O.C.*, vol. I, pp. 222–226, November 2002.

[9] T. Salonidis, P. Bhagwat, and L. Tassiulas, "Proximity awareness and fast connection establishment in Bluetooth," *Mobile and Ad Hoc Networking and Computing*, pp. 141–142, 2000.

[10] C. C. Foo and K. C. Chua, "BlueRings - Bluetooth scatternets with ring structures," *IASTED International Conference on Wireless and Optical Communication (WOC 2002), Banff, Canada*, July 17-19 2002.

[11] M. A. Marsan, C. F. Chiasserini, A. Nucci, G. Carello, and L. D. Giovanni, "Optimizing the topology of Bluetooth wireless personal area networks," *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings.IEEE INFOCOM 2002*, vol. 2, pp. 572–579, June 2002.

[12] T.-Y. Lin, Y.-C. Tseng, and K.-M. Chang, "Formation, Routing, and Maintenance Protocols for the BlueRing Scatternet of Bluetooth," *36th Hawaii Intl Conf. on System Sciences (HICSS) - Track 9*, p. 313a, January 2003.

[13] C. Law and K.-Y. Siu, "A Bluetooth scatternet formation algorithm," *IEEE Global Telecommunications Conference, 2001. GLOBECOM '01*, vol. 5, pp. 2864–2869, November 2001.

[14] C. Law, A. Mehta, and K.-Y. Siu, "A New Bluetooth scatternet formation algorithm," *ACM Mobile Networks and Applications Journal*, vol. 8, no. 5, pp. 485–498, October 2003.

[15] G. Tan, A. Miu, J. Guttag, and H. Balakrishnan, "An Efficient Scatternet Formation Algorithm for Dynamic Environments," *IASTED Communications and Computer Networks (CCN), Cambridge, MA*, no. 0-88986-329-6, November 2002.

[16] G. Tan, "Self-organizing Bluetooth Scatternets," *SM Thesis, Massachusetts Institute of Technology*, January 2002.

[17] S. Basagni and C. Petrioli, "A Scatternet Formation Protocol for Ad hoc Networks of Bluetooth Devices," *IEEE Vehicular Technology Conference*, vol. 1, pp. 424–428, Spring 2002.

[18] J. Yun, J. Kim, Y.-S. Kim, and J. Ma, "A Three-Phase Ad Hoc Network Formation Protocol for Bluetooth Systems," *Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications(WPMC) 2002, Hawaii*, October 2002.

[19] I. Stojmenovic, "Dominating set based Bluetooth scatternet formation with localized maintenance," *Proceedings of the Workshop on Advances in Parallel and Distributed Computational Models, Fort Lauderdale, FL*, p. 148b, April 2002.

[20] P. Enge and P. Misra, "Special Issue on GPS: The Global Positioning System," *Proceedings of the IEEE*, pp. 3–172, January 1997.

[21] M. S. Rohit Kapoor and M. Gerla, "An Analysis of Bluetooth Scatternet Topologies," *ICC 2003, Anchorage, Alaska*, vol. 1, pp. 266 –270, May 2003.

[22] X. Zeng, R. Bagrodia, and M. Gerla, "Glomosim: a library for parallel simulation of large-scale wireless networks," *Proceedings of the 12th Workshop on Parallel and Distributed Simulations – PADS '98*, May 1998.