Bluetrees—Scatternet Formation to Enable Bluetooth-Based Ad Hoc Networks

Gergely V. Záruba, Stefano Basagni, and Imrich Chlamtac
Center for Advanced Telecommunications Systems and Services (CATSS)
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
E-mail: {zaruba,basagni,chlamtac}@utdallas.edu.

Abstract— **Bluetooth is an open specification for short-range wireless communication and networking, mainly intended to be a cable replacement between portable and/or fixed electronic devices. The specification also defines techniques for interconnecting large number of nodes in *scatternets*, thus enabling the establishment of a *mobile ad hoc network* (MANET). While several solutions and commercial products have been introduced for one-hop Bluetooth communication, the problem of *scatternet formation* has not yet been dealt with. This problem concerns the assignment of the roles of *master* and *slave* to each node so that the resulting MANET is connected. In this paper we introduce two novel protocols for forming connected scatternets. In both cases, the resulting topology is termed a *bluetree*. In our bluetrees the number of roles each node can assume are limited to two or three (depending on the protocol), thus imposing low slave management overhead. The effectiveness of both protocols in forming MANETs is demonstrated through extensive simulations.**

1 Introduction

It has been widely predicted that Bluetooth [1] will be the major technology for short range wireless networks and wireless personal area networks (WPAN). Because of the expected low cost of Bluetooth chips (down to 5 USD, according to leading manufacturers), Bluetooth based networks are expected to be the preferred solution to build inexpensive but large *ad hoc networks*, i.e., multi-hop radio networks whose nodes may all be mobile. This paper deals with the problem of building ad hoc networks using Bluetooth technology.

The original idea behind the Bluetooth concept was that of cable replacement between portable and/or fixed electronic devices. According to the standard, when two Bluetooth devices come into each other's communication range, one of them assumes the *role* of *master* of the communication and the other becomes the *slave*. This simple "one hop" network is called a *piconet*, and may include more slaves. There is no limit on the maximum number of slaves connected to one master, although the number of *active slaves* at one time can not exceed 7. If a master has more than seven slaves, some slaves have to be *parked*. To communicate with a parked slave a master has to "unpark" this while possibly parking another slave.

The standard also allows multiple roles for the same device: for instance, a node can be a master in one piconet and a slave in one or more other piconets. This permits the connection of several piconets: The nodes with multiple roles will act as gateways to adjacent piconets. In the Bluetooth terminology the network topology resulting by the connection of piconets is called a *scatternet*. Theoretically, Bluetooth does not pose limits on the number of roles that a node can assume. The only restriction is that at one time one node can be active only in one piconet. To operate as a member of another piconet, a node has to be able to switch to the hopping frequency sequence of the other piconet. Since each switch costs in terms of switching delay, scheduling, re-synchronization, etc., an efficient *scatternet formation* protocol can be one that minimizes the roles assigned to the nodes, without losing network connectivity. (For a more detailed overview on Bluetooth the reader is referred to [3] or directly to the Bluetooth specification [1]).

The problem of organizing ad hoc networks with Bluetooth devices comes from the Bluetooth specification itself: even though two nodes may be physically able to receive each other transmissions, they cannot communicate if they are not in the same piconet at the same time (one assuming a master the other a slave role). Thus with respect to the generic ad hoc technology, where two nodes can communicate if their distance is less than their transmission radii, in building Bluetooth scatternets attention has to be paid to choosing masters and slaves so that the network connectivity is maintained.

A scatternet is visualized in Figure 1. Given the network topology graph obtained according to the generic ad hoc network technology, a direction is assigned to those links, that represent an established master slave relation. The arrows point from masters to slaves: A node that has an out-degree of $k \geq 1$ is the master of a piconet consisting of k slaves. A node with an in-degree of $m \geq 1$ is a slave in m piconets. The directed links are the only links that will be used in the Bluetooth scatternet, i.e., non-directed links in the figure will not be available for communications, namely, some of the network connectivity is lost. (Note that the direction on the edges does not define a one-way communication: it is only used to depict master-slave

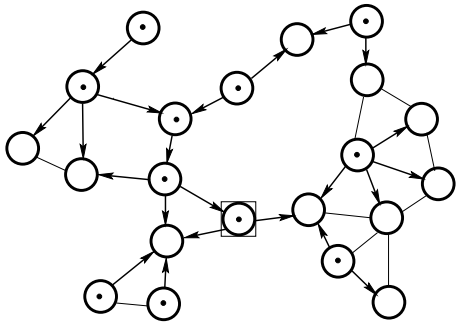

Figure 1: A Scatternet.

relations among nodes.)

The problem of Bluetooth scatternet formation has the primary aim to decide about the roles among the Bluetooth devices so that the resulting scatternet enables multihop communication between two arbitrary devices. A goal in different scatternet formation algorithms is to minimize the overheads introduced by Bluetooth's piconet and scatternet based master slave approach. This problem has not been investigated in depth so far. In general, most of the papers published about Bluetooth only consider piconets (see, e.g., [3]), and the problem of scatternet formation is not addressed at all. Scatternet formation has been briefly illustrated in [6], where the authors consider randomly generated topologies of Bluetooth nodes and investigate the trade-off between routing efficiency and link numbers. They do not consider piconet switching times, while the random selection does not have a constraint on the number of roles assumed by nodes.

In this paper we introduce two protocols for scatternet formation considering role and link limitations. Both solutions are tailored to Bluetooth functionality and are distributed, i.e., they are executed at each node based only on the knowledge of the node's immediate neighbors.

The first algorithm is based on a designated node, the "blueroot," that initiates the construction of a "bluetree," i.e., the resulting scatternet topology will be a tree that spans the entire network. In this solution, the number of roles assigned to one node is limited to two, i.e, a node can be only a master, a slave, or the slave in two different piconets or a master in a piconet and a slave in an adjacent piconet. This protocol is also extended to cope with a limitation on the number of slaves controlled by one master to reduce parking related overhead. Simulation results of networks with up to 2000 nodes show the relation of the obtained *blueroutes* with respect to the optimal (shortest path based) case evaluated on the network topology graph.

The second algorithm speeds up the scatternet formation process by selecting more than one root for tree formation, and then merging the trees generated by each root. The protocol is organized in two phases. In the first phase, a subset of the nodes will be selected as *init nodes* that initiate the construction of subtrees, similarly to the first algorithm. We show via simulation that the set up time for the sub-trees has a strong dependence on the network density, and is not significantly influenced by the network population. In the second phase, the protocol merges the generated sub-trees into one scatternet that spans the entire network.

The paper is organized as follows. In Section 2 we describe the proposed algorithms in details, while Section 3 presents the simulation results. Finally, Section 4 concludes the paper.

2 Bluetrees–Scatternet Formation

In this section we detail the two protocols for the formation of Bluetooth scatternets from a networks of Bluetooth nodes that are placed randomly in a given geographic region. We consider networks with low node mobility (coping with mobility is the subject of on-going and future research).

Both solutions assume that each node, upon terminating its boot procedure, is aware of the number and identities (i.e., the Bluetooth address) of each of its neighbors. Thus, during the boot process, nodes spend enough time inquiring and responding to inquiries to discover all their neighbors.

Throughout the rest of the paper, by "geographically connected" network, we intend the networks obtained according to the generic ad hoc technology, where two nodes are neighbors if that they are within each other transmission range, and there is at least one path between any two nodes in the network. We use this definition to compare our solutions to the solution obtained with such a generic technology.

2.1 Blueroot Grown Bluetrees

Theorem 1 (Bluetooth Trees) *Every network that is geographically connected admits at least one corresponding connected Bluetooth scatternet, where the number of roles assigned to nodes are limited to at most two per node (i.e., each node is one of* $\{M, S, (MS)\}$*).*

The proof of this theorem comes from the algorithm described below:

The first protocol is initiated by a given, single node, called the the *blueroot*, which will be the root of the *bluetree*. The protocol is described in the following way.

Let us select an arbitrary node t (i.e., the blueroot). Using the network topology graph, we build a rooted spanning tree with root t. The root will be assigned the role of master (M). Every one-hop neighbor of t will be a slave (S) in the piconet of t. This is depicted by drawing a directed link from t to its "children." The children of t will now be assigned an additional role M and all their neighbors that are not assigned any roles yet will become slaves S of these newly selected masters. This procedure is repeated recursively till the "leaves" of the tree are reached (these nodes will only be slaves). We stipulate that when a node which has not been assigned a role yet and is the neighbor of more than one master, it affiliates with the master whose page reached it first. In this scenario, each node can only

have one M role, two S nodes can not be directly connected to each other, and all S and MS nodes have only one master that controls them.

Scatternets formed by following the proposed algorithm can be depicted as directed trees rooted from the given blueroot. A bluetree corresponding the the topology graph of Figure 1 is depicted in Figure 2 (the squared node is the blueroot).

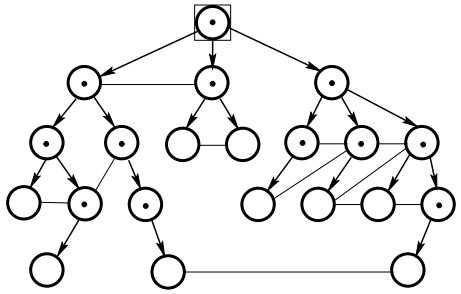

Figure 2: A Rooted Bluetree

In the following we describe the formation rule stated above according to the Bluetooth specification.

2.1.1 Bluetree Formation

It is assumed that each node knows: 1) whether or not it is the blueroot, 2) what are the identifiers of its one hop neighbors, and 3) whether they are part of a piconet already.

If a node is the blueroot, it starts paging its neighbors one by one. This implies that the blueroot will be a master. If a node is paged and it is not part of any piconet yet, it accepts the page thus becoming the slave of the paging node. Otherwise, it will either not respond to the page or it may respond and immediately after having informed the paging node that it is "already taken" it disrupts the communication. This procedure is recursively repeated till all nodes are assigned to a piconet: once a node has been assigned the role of slave in a piconet, it initiate paging all its neighbors one by one, and so on.

2.1.2 Properties of Bluetrees

Most of the distributed algorithms for finding a spanning tree in a network, whether wired or wireless [2] work by creating tree segments over the vertices and expanding these segments (possibly by interconnecting them) until the whole set of vertices is spanned by one tree. Since most vertices do not have the knowledge on which edge leads to a common root vertex, they cannot establish the correct master-slave setup (the edges cannot be directed in a top-down way.

Routing in rooted bluetrees can be performed in the following way. Once a node realizes that it is a leaf (only slave role), it informs its master about it. Thus, masters can maintain next-hop routing tables with entries of all the descendants in the corresponding subtree. The routing table of a master is then sent "upstream" on the tree to the master's master, which enlarge the table with information related to its own children and pass it to its master, and so on till the blueroot is reached. When routing packets, nodes examine their routing tables. If the destination is in their table, then the packet is routed towards the corresponding slave, otherwise the packet is forwarded to upward, to this node's master. Since the blueroot has information about all the nodes in the network the packet will eventually reach its intended destination.

2.2 Limiting the Number of Slaves

By building the bluetree as described above, it can happen that a master is assigned too many slaves. Since the a master can only control 7 active slaves, this can introduce excessive overhead and delays. In this section, by using a simple geometric observation we reconfigure the obtained bluetree so that it satisfies any limit in the number of slaves greater than 5.

Observation: In an open, interference- and obstacle-free environment, if a node n has more than five neighbors, then there are at least two nodes among these neighbors that are neighbors themselves.

The geometric basis of our observation is the following. If a node n has more than five neighbors, let us arbitrarily select six among them. The "worst case" placement of the nodes is when they form a perfect hexagon on the edge of the transmission radius r of node n. In perfect hexagons adjacent corners are of the same distance r as corners from the middle point. Since the transmission radius of all nodes is r, the nodes in the hexagon next to each other are neighbors.

This observation can be used to reconfigure the bluetree so that each master has no more than $L > 5$ slaves. The following algorithm is executed at each node until all nodes satisfy the slave constraint. If a master m has more than L slaves, then it can be sure that at least two of this slaves could have a possible link between them. The master can poll the slaves to find out the identifiers of their neighbors, and to find out how many slaves they handle themselves. Using this information the master can select two of its slaves s_1, and s_2 so that they can be connected. The selection can also consider s_1 as the node that has the fewest number of slaves from the applying set. Then s_1 can be instructed to establish connection with s_2 being its master (m can provide the clock offset to speed up paging), while s_2 is instructed to discontinue the connection to m and wait for the page of s_1. After each such branch reorganization step the resulting topology retains the bluetree properties and all masters will have at most L slaves.

As it is clear form the above description, the proposed protocol and its variation are very simple, adapt naturally to Bluetooth constraints, incur minor overhead both for local computation and transmissions, and produce a scatternet that is ready to be use as a backbone to disseminate information throughout the network.

2.3 Distributed Bluetrees

In this section we further distribute the tree formation protocol, saving on set-up time and avoiding the a priory designation of the blueroot. As for the protocol described above, this protocol achieves network connectivity while maintaining a limited number of roles per node. In particular, each node has roles assigned from the set $\{M, S, (MS), (SS), (MSS)\}$.

The first step in our algorithm is to select *init nodes* among the population in a distributed manner. To select the init nodes, the information on whether or not a node has the highest identifier in its neighborhood can be used: if a node is surrounded only by smaller id nodes, it elects itself as an init node. The init nodes then initiate the bluetree building procedure described in the previous sections, with the following modifications: 1) when a piconet connection is established, the slave will be informed about the identifier of the root of the tree; 2) when paging neighboring nodes which are already part of a bluetree, information on respective roots has to be exchanged. The information collected by these two modifications will be used in the second phase of the algorithm. At the end of the first phase, the network topology graph will be spanned by disjunct but adjacent trees, with each node having roles from the set: $\{M, S, (MS)\}$.

For the second phase a procedure is sought, that connects these sub-tree scatternets into one scatternet spanning the entire network topology graph. The same algorithm needs to ensure that no node will violate the number of roles constraint. We designate one of the scatternets as the root of the final phase. We can map each sub-tree by a node in a virtual graph and possible edges between sub-trees as edges in this graph. We can run the bluetree algorithm on this virtual graph, to connect all virtual vertices. In the real graph that would result in a connected network. Also, nodes in this virtual graph could only receive an additional M, S or MS role, so the final role set could be calculated by multiplying this set with itself, resulting in: $\{M, S, (MS), (SS), (MSS)\}$ (note, that $(MM) = M$). The information collected in the first phase can be used to decide on the links to be activated between sub-trees. Data exchange in the sub-trees becomes necessary in order to handle these sub-trees as virtual vertices. s With this distributed bluetree procedure, we pay the increased robustness and distribution in terms of overhead in the second phase. Figure 3 shows a possible scatternet created with the second algorithm. The squared nodes represent highest identifier nodes in their respective neighborhood, and outlined arrows depict master-slave behavior established by the second phase.

3 Simulation Results

Our C++ based simulator operates on graphs representing networks. Great effort has been put fort matching our simulator to the expected behavior of Bluetooth units (e.g., the BFST algorithm is not implemented synchronously but has random distributed behaviors). The first step in all our simulations was to

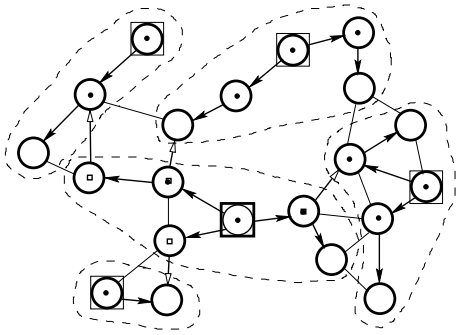

Figure 3: Combined Distributed Bluetrees

create geographically connected random networks by spreading nodes uniformly over an area limited by coordinates X_m and Y_m, while keeping the "transmission radius" of nodes constant. To obtain results for different network populations N, and different densities D, the parameters X_m, Y_m and N are used as simulation inputs.

We show results on the average distances (or routes) of the first bluetree algorithm (Br) compared to the minimal average distances(Or). The minimal distances were calculated using a shortest path algorithm on the network topology graph. The Br and Or values are used to calculate average route errors $ARE = Br - Or$ between nodes. Figure 4 displays ARE plots for bluetree scatternets, with no limit on the number of slaves connecting to a master. As it can be predicted, forcing a limit on the number of slaves will result in higher ARE values.

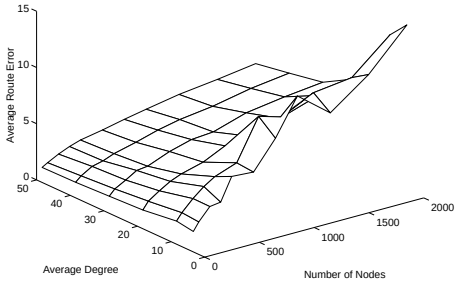

Figure 4: Average Route Error (No Limit)

Figure 5 plots ARE values when the number of slaves is limited to 7 slaves. In our simulations ARE curves were almost identical for limits 5 to 7, thus figures on lower limits are omitted. Comparing the curves to the not limited scenario one can observe a significant change ARE: the ARE can be twice as much in a limited case, while the function surface changes direction and increases radically with the average nodal degree. Limiting the slaves results in a stronger dependence on the population.

In the distributed bluetree algorithm it is obvious that the

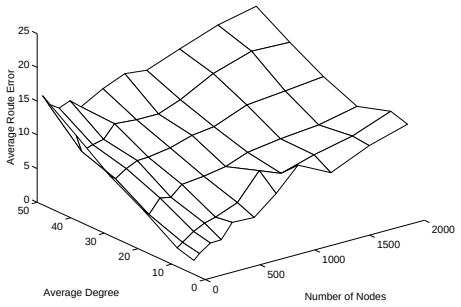

Figure 5: Average Route Error (Limit = 7)

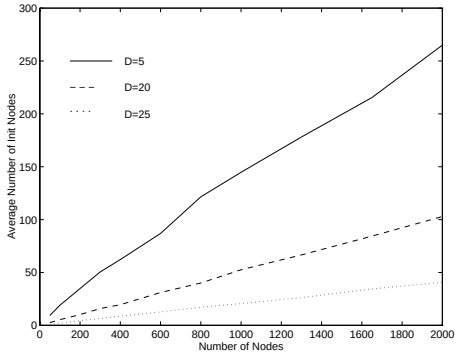

Figure 7: Average Number of Init Nodes - 2D

speed up factor is proportional to the number of init nodes, i.e., the more init nodes among the population, the more distributed the algorithm will be. We show simulation results on how many init nodes there are in the networks. Figure 6 gives a surface view on the shape of such a curve with the population and the density on the x and y axes. In Figure 7 actual values can be read for different network sizes with nodal degrees of $5, 20, 50$. The curves in Figure 7 are quasi-linear in N with zero offset, implying that in an average network the duration of the first phase will not be significantly influenced by the total number of nodes, just by the network density. With other words, in average, the duration of the first phase of the algorithm will be the same no matter how big the network is. Another observation can be made by looking at the influence of D to the curves: it will be always more time efficient to run the algorithm's first step on a sparse network. Although the second phase is expected to be faster if the network is more dense. In large networks, the formation time will be dominated by the run time of the second phase of the algorithm. Since in the second phase the bluetree algorithm operates on trees and not nodes like in the first algorithm, the second algorithm is expected to be faster for large networks, with a factor proportional to the average subtree size. The average subtree size is the inverse of the curve slopes in Figure 7.

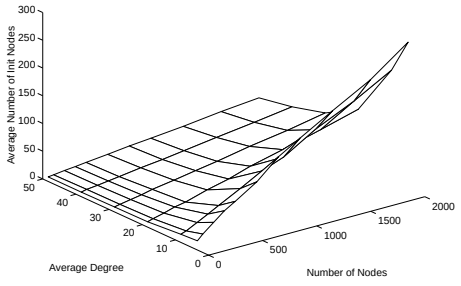

Figure 6: Average Number of Init Nodes - 3D

All our simulations have been completed with confidence levels above 95%, while keeping the error margin below 5%.

4 Conclusions

The Bluetooth standard enables the formation of scatternets, with which mobile ad hoc networks (MANETs) may be established. In this paper we presented two protocols to solve the problem of scatternet formation. To the best of the authors' knowledge, these are the first two attempts proposed for solving this problem.

The first protocol organizes the Bluetooth nodes into a rooted tree, the bluetree, in which each node participates at most two piconets, reducing piconet switching overhead. Our solution also considers the problem of limiting the number of slaves that each master has to control. Simulation results in this case show that the average route ratio is dominated by the density and not significantly influenced by the population size. We presented a second protocol with a more distribute behavior. Using simulation we have demonstrated that in a large population of nodes the speed of the formation procedure outperforms that of the first one. In the second solution the piconet switching overhead is still well contained, since each node is allowed to assume at most three roles.

References

[1] Bluetooth SIG, *Bluetooth Baseband Specification Version 1.0B*, http://www.bluetooth.com

[2] R. Gallager, P. Humblet, P. Spira *A Distributed Algorithm for Minimum Weight Spanning Trees*, ACM Transaction on Programming Language and Systems, pp. 66-77, Vol. 4, No. 1, January, 1993

[3] J.C. Haartsen, *The Bluetooth Radio System*, IEEE Personal Communications, pp. 28-36, Vol.7, February, 2000

[4] Homepage of the IEEE 802.15 WG, *Working Group for Wireless Personal Area Networks*, http://grouper/ieee.org/groups/802/15/

[5] Internet Engineering Task Force, *MANET WG Charter*, http://www.ietf.org/html.charters/manet-charter.html

[6] Gy. Miklós, A. Rácz, Z. Turányi, A. Valkó, P. Johansson, *Performance Aspects of Bluetooth Scatternet Formation*, poster section of MobiHoc 2000, Boston, August 2000