

Energy-Efficient Bounded-diameter Tree Scatternet for Bluetooth PANs

Muralidhar Medidi
Jonathan Campbell
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752
{mmedidi, jcampbel}@eecs.wsu.edu

Abstract

Bluetooth is a promising wireless technology that enables devices to form short-range multihop wireless ad-hoc networks, or personal area networks (PANs). However, scatternet formation is one of the challenges that need to be resolved since the performance of a Bluetooth network depends largely on the scatternet topology used. We first identify a particular variant of a height-balanced binary tree, termed ACB-tree for almost-complete-binary tree, that allows two such trees to be combined to create a larger ACB-tree while retaining the height-balance requirement. We then present a distributed scatternet formation algorithm for creation of ACB-trees. We further extend the algorithm to produce an ACB-tree scatternet with energy efficient properties. We also present simulations, conducted using Blueware simulator, to provide experiment results to study and compare the performance of the resulting scatternets.

1. Introduction

Bluetooth is a low-cost and low-power short-range radio technology, intended to replace cable(s), connecting portable and/or fixed electronic devices. It operates in the unlicensed 2.4 GHz ISM (Industrial, Scientific and Medical) band and uses frequency hopping scheme to avoid interference from other devices and wireless technologies (e.g. 802.11) operating in the same band.

When two Bluetooth devices establish a connection, one of them is assigned the role of a *master* of the connection and other becomes its *slave*. The resulting simple one-hop network is called a piconet. There is no limit on the maximum number of slaves in a piconet; however, the number of *active* slaves in a piconet is always limited to 7 (Bluetooth specification 1.1, [1]) and usually referred as k . If a master has more than k slaves, extra slaves must be *parked*.

Piconets can be interconnected into a *scatternet* by sharing slaves, termed *gateway nodes* or *bridges*.

Essentially, the scatternet formation problem is the assignment of roles to Bluetooth devices or nodes. This assignment, which determines whether a node plays role of a master, slave or bridge, also determines structure of the scatternet, and hence, greatly affects the overall throughput [12].

Initially, Bluetooth devices have no knowledge of their surroundings or other devices. Each device operates independently of others and hence, a *distributed* scatternet formation scheme must be used. Devices can be mobile, so topology changes are expected frequently. The scatternet topology should be dynamic, meaning that it should allow for dynamic addition and deletion (leaving) of nodes. Also, scatternet should be formed within a reasonable time.

In order to improve the performance of the scatternet, following are some important criteria that must be considered: (1) the number of piconets in the scatternet should be kept low, (2) low degree of a node, and (3) low network diameter. It is also desirable to create a scatternet which is mindful of energy efficiency during topology construction to allow the network to exist for the longest period possible without re-organization. Though it has been shown that the tree structure has a bottleneck at the root under uniform traffic loads [23], this property creates a natural concept for placement of nodes within the tree with respect to energy efficiency.

In this paper, we study the problem of energy-efficient scatternet formation for devices which are all within transmission range of each other: we identify a height-balanced binary tree structure, ACB-tree, that allows easier combination and then propose a distributed scatternet formation technique to create these ACB-trees. The rest of the paper is organized as follows. Section 2 gives a brief overview of the Bluetooth specification as it relates to scatternet formation. Related research on scatternet formation is summarized in Section 3. We first present the ACB-tree and describe ATSF, the ACB-tree scatternet formation, algorithm

in Section 4. Then, we discuss an energy-efficient modification to the standard ACB-tree scatternet formation algorithm in Section 5. In Section 6, we present simulation results and compare them with those available for other scatternet formation algorithms. We provide some concluding remarks in Section 7.

2. Background

The basic building block of the Bluetooth network is a *piconet* which is made up of a master node and a number of slaves. All devices in one piconet share the same communication channel, using the frequency hopping sequence determined by the master, and can communicate only through the master node.

According to Bluetooth specifications, the link formation process consists of two major phases: *inquiry* and *page*. A Bluetooth device can discover other neighboring devices by the *inquiry* process. Nodes randomly choose to be in INQUIRY or INQUIRY SCAN state: those in INQUIRY start looking for neighbors which are in the INQUIRY SCAN and nodes in INQUIRY SCAN are waiting to be contacted. A full description of the link formation process can be found in [1] or [6].

If the inquiry phase is successful, the INQUIRY node acquires the information about the node in INQUIRY SCAN and then, both nodes move into PAGE and PAGE SCAN respectively. This starts the *page* process: the node in the PAGE becomes the *master* and node performing PAGE SCAN becomes its *slave*.

A single Bluetooth device is allowed to be a part of several piconets acting as a gateway between them. Using these gateways, we can interconnect independent piconets into a single, larger multi-hop network or personal area network (PAN); in Bluetooth terminology, such a network is termed *scatternet*. Though Bluetooth specifications identify methods for device discovery and also allow a node to participate in more than one piconet, scatternet topologies and their formation is unspecified and left upto the developers.

3. Related Work

Scatternet formation for Bluetooth networks has been an active area of research. The proposed algorithms can be roughly divided into two classes: those that assume all devices are within transmission range of each other and those that aim to form a connected network when all devices are not necessarily in range of each other. Our algorithm falls in the first category as we assume all devices are in transmission range of each other. The algorithms suggested for out-of-range devices include [15, 16, 19, 22]. Interested readers are referred to [6] for a discussion about scatternet formation for out-of-range devices.

In the in-range category, initial attempts included a scatternet formation algorithm given by Salonidis *et al.* [18] called *Bluetooth Topology Construction Algorithm* (BTCP). With the aim of complete connectivity, every piconet is connected to every other piconet through a slave/slave gateway. The generated scatternet can support a maximum of 36 nodes and does not scale to a higher number of nodes. Ramachandran *et al.* [17] redefined the problem of scatternet formation as a problem of *clustering*. However, the resulting scatternet is not guaranteed to be connected since the problem of determining gateways between two piconets is not addressed.

Law and Siu [10] proposed a single phase decentralized scatternet formation algorithm with a basic focus on minimizing the number of piconets and degree of a node in the network. The resulting scatternet is a tree, with minimal number of bridge nodes, reducing the synchronization delay and hence overall end-to-end delay. Tan *et al.* [21] suggested TSF, a self healing tree, accommodating topology changes and reorganizing to retain the structure. Chong and Chaing [8] created Bluering, a scatternet with ring structure. Zhang, Hou and Sha [23] proposed formation of *loop scatternets* and also identified the node contention as a performance metric for the scatternets. Helttunen, Mishra, and Park [9] proposed a method for merging piconets to find common bridge nodes. Baatz *et al.* [4] proposed a scatternet formation algorithm based on 1-factors.

Barriere *et al.* [5] suggested a distributed scatternet formation algorithm for creation of so called *projective scatternets*. Sunkavalli and Ramamurthy [20] proposed MTSF, a mesh formation scheme which focuses on reducing scatternet formation time and fast dynamic node acceptance times. Medidi and Daptardar [11] suggested BlueMesh, a distributed algorithm which produced a regular mesh scatternet to reduce the number of piconets and the diameter. Persson and Manivanan [14] proposed a distributed self-healing protocol which was shown to be fault-tolerant and allowed for multi-hop formation, but had no control over the shape of the resulting scatternet. Chen *et al.* [7] proposed a method for creating scatternets where the resulting topology could be controlled by manipulating a set of constraints, such as the maximum number of loops or the maximum number of slaves. It should be noted that results were only shown for a very small number of nodes (1-8) and the algorithm is not a distributed one.

The only scatternet formation algorithm available which targets energy efficiency as a metric is SF-DeViL [13]. Each device in the scenario has a device grade, which is derived from the class of device and the battery level. This class is defined by grouping together devices which have similar energy capabilities. In addition to the device grade, each device assigns a received signal strength grade for each of its neighbors indicating the received signal as weak,

medium, strong or very strong. By using these two values, a 'best master' is selected among a piconet. The scatternet is formed in a basic tree fashion and then reorganized such that the 'best master' among each piconet is promoted to be the master. For example, if a cellular phone is initially the master of a piconet and many or all slaves are laptops, one of the laptops would be selected as the best master to reduce the load on the cellular phone. At the end of the formation process, the final topology consists of a single spanning tree with the most energy capable device as the root and the least capable devices as leaves. The simulation results presented indicated that this scheme has a high formation delay in comparison to other scatternet formation algorithms. However, none of the other algorithms considered energy efficiency as a metric.

4. ACB-tree and Scatternet Formation

Our primary objective is to obtain a scatternet of bounded diameter: binary trees, if height-balanced, provide a minimalist interconnection network with logarithmic diameter and with a low degree at any node. Further, scatternet formation which connects independent Bluetooth devices has to be fundamentally distributed in nature to provide scalable operations. However, traditional trees (for example: full or complete binary trees) with bounded heights defy easy distributed formation.

Based on the characteristics of Bluetooth link discovery and distributed formation objective, we identified a binary tree structure that serves these purposes. ACB-tree (for almost-complete-binary tree) is a complete binary tree with an additional node connected to the root, illustrated in Figure 1. Recall that a complete binary tree of depth d contains $2^{d+1}-1$ nodes. Merging two such complete binary trees into a larger one requires an additional node to act as the new root, at least, and cumbersome in designing a distributed algorithm. On the other hand, an ACB-tree contains 2^{d+1} nodes. In particular, given two such ACB-trees, the combination to grow into a bigger ACB-tree does not require any additional nodes. And, as shown in Figure 2, two ACB-trees can be combined into a larger one by replacing one edge (from a handle node to its root) with two new edges. In Figure 2, edge (x, y) is replaced with two edges: (a, y) and (a, x) . Note that (as opposed to the illustration in the Figure showing the tree combination) even if two ACB-trees of different heights are combined, the height of these trees is logarithmic as long as the difference in the heights of trees being combined is bounded by a constant.

The device discovery process in Bluetooth is asymmetric, with one of the devices becoming a master of the link established and the other a slave. For symmetric and controlled scatternet formation, and to minimize (maximize) the number of piconet (the size of each piconet), we ori-

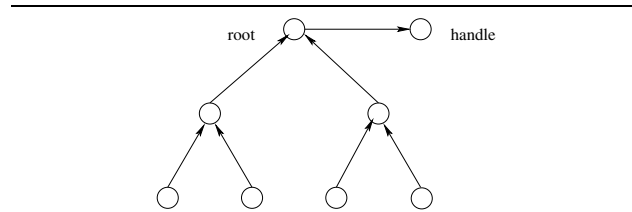


Figure 1. ACB tree

ented the tree links from child nodes to parents (as opposed to the conventional downward pointing tree links). Since the discovery process to grow the tree requires the availability of one free link, and Bluetooth specifications restrict the number of active slaves to seven, each piconet can at most acquire five dedicated slaves. For ease of description later, we term the root of any ACB-tree as the *coordinator* and the handle node as the *leader*.

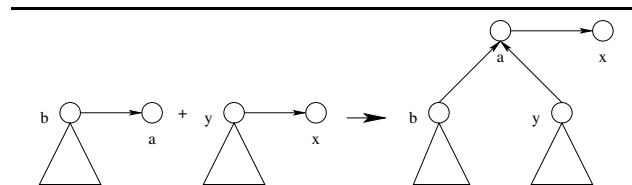


Figure 2. ACB tree combination

Our scatternet formation algorithm ATSF, which generates an ACB-tree, consists of three conceptual phases. In the first phase, all independent Bluetooth devices are connected into piconets, each restricted to a maximum of $k-2$ slaves. In phase 2, all such piconets are glued together to create trees using *recursive doubling* technique. Due to the distributed and controlled nature of the algorithm, to ensure height balance in the tree, the second phase cannot guarantee the generation of a single tree. As a result, in the final phase, we enforce relaxed connections to force the generation of a single, connected tree. Phases 1 and 2 may overlap in time, in the sense that a device u may be in phase 1 while another device v may be in phase 2.

Phase 1 - Piconet Formation: Initially, we assume a set of N isolated devices that are within transmission range of each other: each device is a **leader** of a piconet consisting of only itself. Every leader calls procedure SEEK with probability p and procedure SCAN with probability $(1 - p)$. When a leader executes procedure SEEK, it goes in INQUIRY and tries to acquire one or more slaves. On the other hand, a leader in procedure SCAN enters INQUIRY SCAN and waits to be contacted by another leader. A successful execution path, in the device discovery process, includes two leaders in opposite modes contacting each other after which they enter PAGE (from INQUIRY) and PAGE

SCAN (from INQUIRY SCAN) to create a connection between them. The leaders may merge their piconets into one if it still keeps the number of slaves in the resulting piconet to $\leq (k - 2)$.

A leader is forced to move to the next phase whenever it has $(k - 2)$ slaves in its piconet. However, this condition solely is not enough to move all leaders into phase 2 since it could take an inordinate amount of time for all the leaders to acquire precisely $(k - 2)$ slaves. Hence we force leaders to move to phase 2 with a time-out for phase 1, even if their piconets are sparse.

Phase 2 - ACB-tree Growing: The main aim here is to combine independent piconets together to form trees. Every leader can accommodate a minimum of one more slave in its piconet. Due to the time-out in phase 1, however, every connection between leaders (of single-piconet trees) in phase 2 first tries to merge their piconets into a single piconet as long as the restriction of a maximum of $(k - 2)$ slaves in each piconet is not violated.

Each piconet's leader will start discovering other trees in this phase to merge and grow bigger ACB trees. To speed up the discovery process, two devices in any ACB-tree will be actively participating in merging trees: the leader and the coordinator. Coordinators enter INQUIRY to discover other leaders, and leaders INQUIRY SCAN to be contacted by other coordinators. When a coordinator and a leader discover and establish a link (from coordinator to leader, to be precise, in this master-slave link), they first exchange information about the heights of their respective ACB-trees. If the heights differ by more than one, the newly established link is torn down and these two devices will continue searching for other trees. If the height difference is within one, the additional link needed between the leaders to complete the merger of these two ACB-trees into a larger ACB-tree is established.

At the end of this merger, both coordinators (nodes b and y in Figure 2) retire and become inactive for the purposes of tree-growing. One of the leaders (node a) takes on the role of the coordinator for the new and larger ACB-tree while the other (node x) keeps its role as the leader. Note that the connection establishment and subsequent tearing down between trees of too different heights is an overhead and increases the scatternet formation time. To alleviate this overhead, we relaxed the height difference between combined trees upto one. Thus the height of these relaxed ACB-trees goes up, but still constrained to be logarithmic with the number of nodes (piconets) in the tree. Thus, in phase 2, we are using *recursive doubling* by allowing similar-sized trees to connect to each other forming bigger trees in a controlled fashion. However, this may not guarantee the formation of a single scatternet at the end of phase 2 and we may end up with a forest of ACB-trees of differing heights.

Phase 3 - Combination into a Single Tree: A leader (and its coordinator) in phase 2 is continuously in search of other trees of appropriate height. However, if the device discovery proves to be unsuccessful after a certain time-out period, the leader declares the end of phase 2 and moves on to phase 3. In this phase, we merge such trees into a single scatternet by ignoring the height requirements. In the special case when a tree must merge with a lone piconet (a piconet which has not merged with any other piconet in phase 2), the lone piconet is 'swallowed' by being placed into the tree as the child of a leaf node.

5. Energy Efficiency

It has been shown that under a uniform traffic model, nodes which are closer to the root node of a tree will have a higher load overall [23]. As a result, it is natural to construct the tree such that the most energy capable nodes serve as bridges and are located higher in the tree. These bridge nodes carry a higher load due to the routing of packets, switching between multiple piconets and handling their own traffic. The least energy capable nodes should be assigned the role of slave nodes, where they are only required to handle their own traffic.

During phases one and two of the algorithm, merges are conducted such that the most energy capable nodes 'win' the role of leading the resulting tree. Each device stores an energy capability value, which is used to reflect available battery power and the type of device. The following is a description of the modifications to the scatternet formation algorithm to implement energy efficient ACB-Tree construction.

The algorithm is not dependant on any one method for determining energy capability, and a variety of factors can determine the energy capability of a given node. As such, any method can be used, so long as a relative ranking among nodes can be determined. One such ranking was discussed by Pamuk and Karasan as a part of the SF-DeviL algorithm [13]. The numbers associated with nodes in the ACB-trees in subsequent figures are relative energy capability rankings on a scale of 0-99, with 99 being the most energy capable and 0 being the least.

Phase 1 - Piconet Formation: The standard phase one procedure is used, with one modification. Each time a merge occurs, the two leaders compare their available energy values. The leader with the greater energy capability remains the master of the new piconet. If the slave of the connection is more energy capable than the master, the link will be reversed using the standard Bluetooth protocol. Otherwise, the existing link will be used as is. As a result of this, every piconet ends phase one with the leader as the most energy capable node in its piconet.

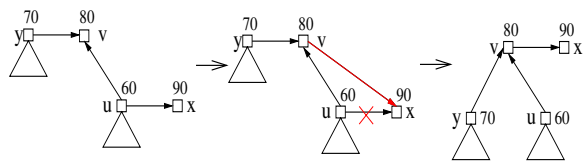


Figure 3. Energy Efficiency - General case of MergeComponents. Existing link is usable.

Phase 2 - ACB-tree Growing: Similar to phase one, the same basic procedure is used for phase two, with the addition of the energy efficiency checking. The same cases exist, but some additional sub-cases are required due to the possible energy ordering of nodes. At the end of phase two, there will be a forest of ACB-Trees, where each tree has the most energy capable node as the leader and less capable nodes will be located further down in the tree from the coordinator node.

The general merge case is illustrated in Figures 3 and 4. The energy levels of node u 's leader (node x) and node v are compared. The node with the higher energy capability is selected to be the leader, while the other node is selected as the coordinator. If x has a greater or equal energy capability than v , the merge process occurs identically to the general case which was described earlier. In the case where v has a greater energy capability than x , the existing u - v link must be disconnected. Before this occurs, v asks its coordinator (node y) to go into PAGE looking for node x , while node u asks node x to go into PAGE SCAN.

If y is successful in paging node x , then the merge process described in the non-energy efficient general case is utilized. Should paging fail, each tree goes back to their respective tasks of searching for other trees.

The result of either sub-case is the same: an ACB-Tree which combines the two smaller trees. The leader of the new ACB-Tree is the most energy capable of the leaders of the two smaller trees, while the coordinator node is the lesser of the two.

There are two special cases which exist due to the possible energy ordering of the nodes. These cases have straightforward implementations which ensure the proper ordering with respect to energy capability.

Phase 3 - Combination into a Single Tree: The most important goal of phase three is to obtain a connected scatternet. As was the case in phase three of the standard algorithm, the height restrictions are relaxed and lone piconets are swallowed by ACB-Trees. The energy efficient procedure used in phase two is used in phase three, while the 'swallowing' technique remains unchanged from the standard algorithm. This provides a good balance between obtaining a connected scatternet in a short period of time while using a best effort technique to unify all remaining trees.

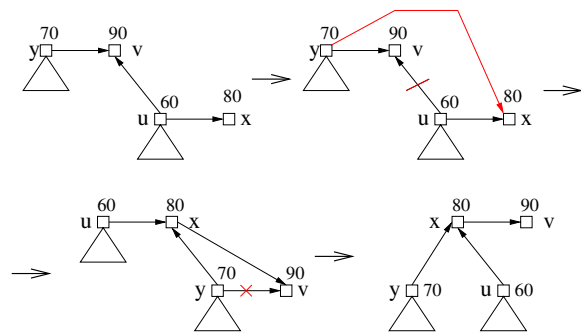


Figure 4. Energy Efficiency - General case of MergeComponents. Existing link is not usable.

At the end of phase three, we will have a single ACB-Tree with the property that, in general, the most energy capable nodes will serve as bridge nodes closer to the coordinator and leader. Those nodes which are less energy capable will be located closer to the leaf nodes or serve as slaves.

6. Simulations

We used Blueware 1.0 [3] for our simulations. Blueware is a Bluetooth extension for NS and closely follows Bluetooth specifications 1.1. It is available from <http://nms.lcs.mit.edu/projects/blueware>. Blueware 1.0 is built on top of the well known network simulator ns-2 [2], which is a discrete-event simulator targeted at networking research and provides substantial support for simulating TCP, routing and multicast protocols for various networking experiments. Blueware simulator implements most aspects of the Bluetooth protocol stack according to Bluetooth specifications 1.1, provides an extensible architecture for various scatternet formation and link scheduling schemes, and to evaluate their performance. The Baseband module implements the pseudo-random frequency hopping technique and several operations as specified in the Bluetooth Baseband. Various implementation challenges have been handled such as the use of *sessions* at the Baseband module. Unlike Bluehoc, another basic Bluetooth simulator, Blueware supports creation of scatternet allowing both master/slave and slave/slave bridges. Although the Bluetooth specification provides the necessary HCI commands to carry out *inquiry* and *page* phases and to activate or hold a communication link, it does not specify scheduling. A separate module called Task Scheduler is provided in Blueware which implements task scheduling framework.

However, there are a few limitations to the Blueware simulator. Blueware currently does not support synchronous connections (SCO). Further, it does not support SNIFF and

PARK modes for a Bluetooth link, yet. Even though the availability of SCO links and low power modes such as PARK would help Blueware more closely follow the Bluetooth specifications and provide even more realistic implementation of Bluetooth specifications, the simulation results obtained are representative of real or prototype implementations. In particular, availability of these additional features would speed up the formation time in our algorithm because we can exploit SCO links to communicate faster during scatternet formation. Similarly, ability to park a few of the slaves will allow each piconet in our phase 1 to collect six slaves instead of the current limit of five and help us further reduce the number of piconets in the scatternet.

Simulation Parameters

We used Blueware 1.0 to create an ad-hoc network of Bluetooth devices, varying the number of nodes as 20, 30, 40, 50, 60, 80, 100, 120, 150, and 200. Each data point reported is an average of results obtained through 10 different simulation runs with varying seeds. The power class 3 Bluetooth (BT) nodes (i.e. nodes with a maximum transmission radius of 10 meters) are assumed to be randomly distributed in an area of 7.07×7.07 meters, ensuring that all nodes would be within range of each other.

Simulation Results

We implemented our distributed algorithm to form ACB-tree scatternets, referred as ATSF in the figures to distinguish it from others, using Blueware 1.0. The energy efficient modification to ATSF is labeled as ATSF-EE. Three other scatternet formation algorithms were used to compare performance characteristics: TSF, BlueMesh, and BlueCube. These algorithms were chosen to provide a breadth of formation shapes (a total of two tree schemes, a mesh scheme, and a cube scheme)

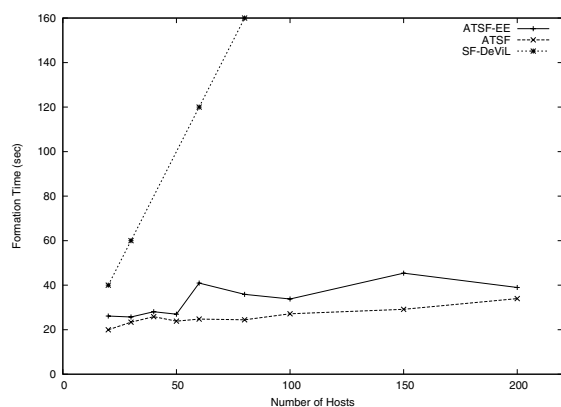


Figure 5. Formation Time vs. Number of Nodes. ATSF-EE vs. ATSF, SF-DeviL.

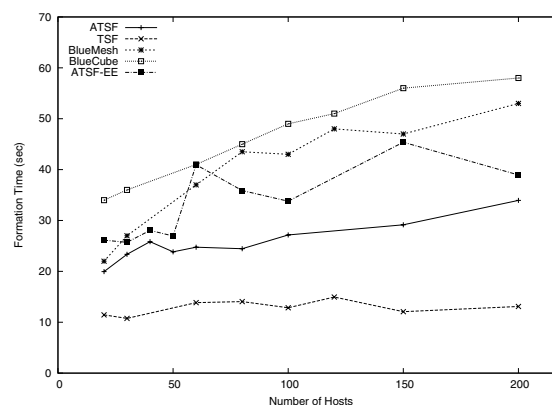


Figure 6. Formation Time vs. Number of Nodes.

and each had a readily-available implementations on Blueware for a fair and direct comparison. Results for SF-DeviL were taken directly from results presented in [13] because an implementation was not available for testing. ATSF-EE produced trees with the same basic properties of the standard ATSF algorithm.

We plotted the scatternet formation times obtained for ATSF, ATSF-EE and SF-DeviL in Figure 5 as the number of nodes or hosts are varied. The results show that the formation time for SF-DeviL increases almost exponentially with respect to the number of hosts. Specifically, a network of only 60 nodes takes two minutes to form. In contrast, ATSF-EE takes less than 40 seconds for the same 60 nodes. In comparison to ATSF, ATSF-EE takes less than 20 additional seconds to form a connected scatternet, with the average cost for obtaining energy efficiency being roughly 7 extra seconds. Note that, as expected, the formation time for ATSF and ATSF-EE increases logarithmically with respect to the number of hosts.

These results correlate with our expectations for ATSF-EE. During the merge process, roughly half of the time the existing links can be used to create a new energy efficient component. In the situation where those links are unusable, only one additional link must be established. This is in sharp contrast to SF-DeviL, which has no bound on the number of links which must be rearranged to obtain energy efficiency.

Figure 6 shows ATSF-EE in comparison to all other tested algorithms. This illustrates that obtaining energy efficiency using ATSF-EE has a low cost when compared to ATSF and the other algorithms. BlueMesh and BlueCube have structures which do not provide a natural energy efficient process for bridge selection. TSF, as expected, requires the least time in scatternet formation, yielding scatternet formation in roughly 12 seconds on average. This is due to the design of TSF, in that it solely focuses on

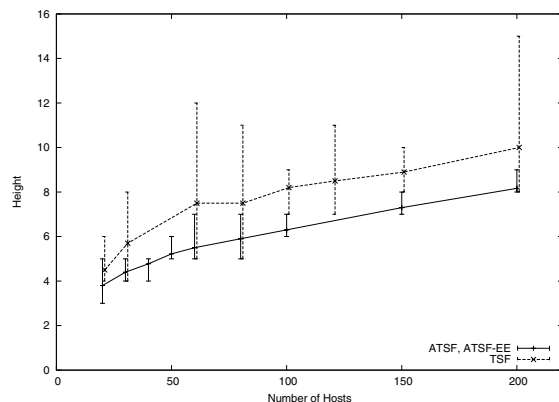


Figure 7. Height vs. Number of Hosts

forming a loop-free connected scatternet. Unlike other algorithms, TSF has no control over scatternet characteristics such as height, diameter, energy efficiency, or the number of piconets.

Figure 7 shows the height of the tree produced by ATSF in comparison to TSF. ATSF-EE produces trees which have almost identical height to those created in ATSF because the energy efficient modifications allow the same merges to occur, only the ordering of the nodes is changed. TSF has the ability to have a maximum fanout of 7, equal to the number of slave nodes, which should produce trees which are much shorter than those produced by ATSF. However, the lack of control over the merge process causes TSF to produce trees that are generally taller than those produced by ATSF. In addition to producing shorter trees, ATSF also produces trees with a diameter which is much more consistent, while the height of the trees produced with TSF varies wildly.

Figure 8 shows the number of battery level violations which exist in the final scatternet. We define a battery level violation as the situation where a node has a higher energy capability than its parent, or when an unshared slave has a higher energy capability than its piconet master.

The number of violations which occur in ATSF increases linearly with respect to the number of hosts in the scatternet. Because the connections are made without regard to energy capability and the energy levels are uniformly distributed, it is reasonable that roughly half of the time the more energy capable node would exceed the less energy capable node. This trend was observed in the experimental data.

ATSF-EE yielded a violation trend which was nearly flat, with only 11 violations occurring with 200 nodes in the scatternet. In comparison, ATSF had 70 violations for the same number of nodes. One cause for these violations is when a lone piconet is swallowed and placed into a hole where the parent has a lower energy capability than the swallowed piconet. The other situation which can create an

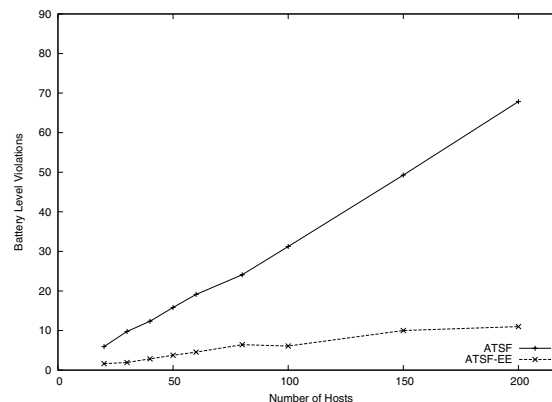


Figure 8. Number of Battery Level Violations vs. Number of Nodes.

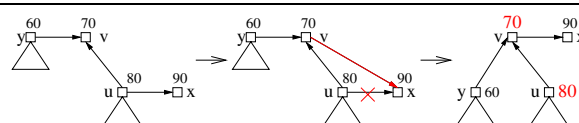


Figure 9. Battery level violation in the general merge case.

battery level violation is illustrated in Figure 9, which is characterized by the leader (node v) of a tree has a lower energy capability than the both the coordinator (node u) and leader (node x) of the other tree. In both of these situations, one battery level violation will result.

While having zero violations would be ideal, a low number of violations could be considered acceptable if the cost for fixing them was too high in terms of the overall scatternet formation delay. At 200 nodes, the scatternet had a 5.5% battery violation rate (the number of violations divided by the total number of nodes). These violations indicate that a more energy efficient structure could be produced, at the expense of increasing the scatternet formation delay.

ATSF-EE maintains all of the properties of the standard ATSF algorithm. The piconet count for ATSF is consistently the lowest among TSF, BlueMesh, and BlueCube. The piconet density is relatively high, with the average piconet having 4.7 nodes. The number of roles for nodes in ATSF is low and fixed, while TSF has no control over the number of roles per node. While ATSF has only a portion of bridge nodes serving multiple roles, TSF requires any node which is not a leaf to serve at least two roles.

Overall, the results for ATSF-EE show that it is feasible to obtain a scatternet with energy efficient properties in a short period of time. The addition of the energy efficient consideration does not invalidate any of the standard

properties of the ATSF algorithm. ATSF-EE is able to produce tree scatternets with bounded diameter, a fixed number of roles per device and fast formation time while adding the additional benefit of energy efficiency.

7. Conclusion and Future Work

We proposed an energy efficient modification for a height-balanced tree, termed ACB-tree, which is suitable for Bluetooth scatternets, and presented a distributed scatternet algorithm to form these trees. Simulation results showed that the ACB-tree scatternets provide fast formation time, and that adding energy efficiency as a metric incurs only a moderate delay increase. In addition, ATSF-EE ensures that the node with the most energy in any piconet serves as the master. The final tree maintains the properties of the standard ATSF algorithm, including a bounded diameter, low number of piconets, and a fixed number of roles per node. The tree maintains the energy efficient structure with few violations.

For future work, our initial efforts for improvement of the ATSF-EE algorithm would be to reduce the number of battery level violations. The amount of time in SEEK or SCAN during phase one could be modified to take the energy capability into account, such that higher-energy capability nodes on the overall will end up as the master of connections.

Bluetooth scatternets are expected to efficiently handle dynamic situations by facilitating node leaves and joins. Notice that nodes from the ACB-tree's leaf nodes can leave the scatternet without affecting the structure or the performance. Nodes, which happen to be masters in the piconets which are internal nodes in the ACB-tree, need to be handled as their leaving could leave the scatternet disconnected. Node joins can be handled elegantly with ACB-tree scatternets: note that each node has at least one empty slot for new devices and can easily absorb new nodes. We are currently investigating how to efficiently accommodate new nodes joining and their incremental cost.

References

- [1] Bluetooth Special Interest Group, <http://www.bluetooth.com>
- [2] NS-2 Network Simulator, <http://www.isi.edu/nsnam/ns>
- [3] Blueware Simulator, <http://nms.lcs.mit.edu/software/blueware>
- [4] S. Baatz, C. Bieschke, M. Frank, P. Martini, C. Scholz and C. K  hl, *Building Efficient Bluetooth Scatternet Topologies from 1-factors*, In Proc. of IASTED Intl. Conf. on Wireless and Optical Communications, WOC 2002.
- [5] L. Barriere, P. Fraigniaud, L. Narayanan and J. Opatrny, *Dynamic Construction of Bluetooth Scatternets of Fixed Degree and Low Diameter*, In Proc. of 14th Annual ACM-SIAM Symposium on Discrete Algorithms, 2003.
- [6] J. Campbell, *Energy-Efficient Bounded-Diameter Tree Scatternets for Bluetooth Networks*, M. S. Thesis, Washington State University, 2005.
- [7] H. Chen, T. Sivakumar, L. Huang and T. Kashima, *Topology-Controllable Scatternet Formation Method and Its Implementation*, Intl. Workshop on Wireless Ad-Hoc Networks, 2004.
- [8] F. C. Chong, C. K. Chaing, *Bluerings - Bluetooth Scatternets with the ring structure*, In Proc. of WOC, 2002.
- [9] J. Helttunen, A. Mishra, and S. Park, *Improved Bluetooth Network Formation (IBNF)*, In Proc. of 27th Annual IEEE Conf. on Local Computer Networks, pp. 304-314, 2002.
- [10] C. Law and K. Siu, *A Bluetooth Scatternet Formation Algorithm*, In Proc. of IEEE Symposium on Ad Hoc Wireless Networks, 2001.
- [11] M. Medidi and A. Daptardar, *A Distributed Algorithm for Mesh Scatternet Formation in Bluetooth Networks*, In Proc. of Intl. Conf. on Wireless Networks (ICWN), pp. 295-301, 2004.
- [12] O. Miklos, A. R  cz, Z. Tur  nyi, A. Valk   and P. Johanson, *Performance Aspects of Bluetooth Scatternet Formation*, First Annual Workshop on Mobile and ad-Hoc Networking and Computing (MobiHoc), 2000.
- [13] C. Pamuk and E. Karasan, *SF-devil : Distributed Bluetooth Scatternet Formation Algorithm based on Device and Link Characteristics*, In Proc. of 8th IEEE Intl. Symposium on Computers and Communication, 2003.
- [14] K. Persson and D. Manivannan, *Distributed Self-Healing Bluetooth Scatternet Formation*, In Proc. of ICWN, pp. 325-334, 2004.
- [15] C. Petrioli, S. Basagni and I. Chlamtac, *Configuring Blues-tars: Multihop Scatternet Formation for Bluetooth Networks* IEEE Trans. on Computers, vol. 52, pp. 779-790, 2003.
- [16] C. Petrioli and S. Basagni, *BlueMesh: Degree-Constrained Multihop Scatternet Formation for Bluetooth Networks*, Mobile Networks and Applications, vol. 9, no. 1, 2004.
- [17] L. Ramachandran, M. Kapoor, A. Sarkar and A. Aggarwal, *Clustering Algorithms for Wireless Ad-Hoc Networks*, In Proc. of 4th Intl. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, 2000.
- [18] T. Salonidis, P. Bhagwat, L. Tassiulas and R. LaMaire, *Distributed Topology Construction of Bluetooth Personal Area Networks*, In Proc. of IEEE INFOCOM, 2001.
- [19] I. Stojmenovic, *Dominating Set Based Bluetooth Scatter-net Formation with Localized Maintenance*, Intl. Parallel and Distributed Processing Symposium, 2002.
- [20] S. Sunkavalli and B. Ramamurthy, *MTSF: A Fast Mesh Scatternet Formation Algorithm for Bluetooth Networks*, Globecom, 2004.
- [21] G. Tan and J. Gutttag, *An Efficient Scatternet Formation Algorithm for Dynamic Environments*, IASTED Intl. Conf. on Communications and Computer Networks, 2001.
- [22] G. Zaruba, S. Basagni and I. Chlamtac, *Bluetrees- Scatternet Formation to Enable Bluetooth-based Ad Hoc Networks*, IEEE Intl. Conf. on Communications (ICC), 2001.
- [23] H. Zhang, C. Hou and L. Sha, *Design and Analysis of a Bluetooth Loop Scatternet Formation Algorithm*, ICC, 2003.