# Forming Connected Topologies in Bluetooth Ad-hoc Networks - An Algorithmic Perspective

R. Guérin, J. Rank, S. Sarkar and E. Vergetis

{guerin@ee, jrank@seas, swati@ee, vergetis@seas}.upenn.edu

Department of Electrical and Systems Engineering, University of Pennsylvania,
200 South 33rd Street, Philadelphia, PA 19104-6390, USA

This paper represents a first step in exploring the formation of connected topologies in ad-hoc networks built on the Bluetooth technology. Connectivity is the most basic requirement for any system aimed at allowing devices to communicate with each other and in this paper we illustrate that this seemingly innocuous goal gives rise to many significant challenges in the context of the Bluetooth technology. We start with a brief overview of Bluetooth and its operation and then identify some of the major problems the technology faces when used to build ad-hoc networks. The paper's contributions are in introducing basic algorithmic problems associated with building connected Bluetooth networks and in developing several possible solutions capable of generating "good" connected topologies.

## 1. INTRODUCTION

Bluetooth is a recently proposed standard for short range, low power wireless communication [15] that was initially envisioned as a wire replacement technology. However, Bluetooth holds the promise of becoming a key enabling technology in allowing the widespread deployment of ad-hoc networks. This is in part because its low power consumption, relatively high bandwidth, and potential low cost make it attractive for the typical mobile devices used in ad-hoc networks.

That being said, there are many significant technical hurdles to cross before Bluetooth can fulfill its potential of becoming more than a wire replacement solution. The most basic challenge that the technology faces, and the one that is the focus of this paper, is how to organize nodes into an operational network while satisfying the many constraints introduced by Bluetooth. There are obviously multiple possible interpretations of what operational means; in this paper we concern ourselves primarily with connectivity.

The paper's contributions are primarily in investigating the problem from an algorithmic perspective. This is a prerequisite to developing a realistic, Bluetooth specific solution. In other words, before forging ahead with the development and implementation of a specific solution, it is critical to first obtain a thorough understanding of the problem space and of the range of possible solutions. We show that the seemingly innocuous problem of

This work was supported in part by NSF grants ANI-9902943, ANI-9906855, ITR-0085930 and ANI-0106984.

deciding whether there exists at least one connected topology that is consistent with the degree constraint of Bluetooth is actually NP-hard in the most general case. However, when node locations are restricted to a two-dimensional plane, we find, under certain simplifying assumptions, that end-to-end connectivity, when feasible, can be achieved with a polynomial complexity algorithm. For general three-dimensional networks we are able to identify algorithms that, when feasible, generate connected topologies most of the time. One of those algorithms actually affords greater control over the topology than mere connectivity. Specifically, it allows independent control of the degree of masters and slaves, respectively. This is important, as bridge connectivity, while not a hard constraint in Bluetooth, can have a major impact on overall network throughput. Another algorithm that we investigate (suggested by Murali Kodialam) is distributed in nature and able to adapt to a dynamically changing topology. These are important properties in practice, even if the resulting topology is not "optimal," or even if the algorithm occasionally fails to construct a connected topology when one exists.

The rest of the paper is organized as follows. We review briefly the salient features of the Bluetooth technology in Section 2. Section 3 is a brief summary of recent research on the topic. Section 4 is devoted to our problem formulation and to showing that the problem of attaining end-to-end connectivity is NP-hard in the general case. Section 5 presents a polynomial complexity topology formation algorithm that, under some simplifying assumptions, results in a connected topology whenever one such exists. In Section 6 we remove those assumptions and present several heuristics that produce good results in this more general setting. Finally, Section 7 investigates the issue of distributed operation and presents a fully distributed and dynamic algorithm.

## 2. CHALLENGES AND OBJECTIVES IN BLUETOOTH TOPOLOGY FORMATION

We first describe the basic features of the Bluetooth technology that are relevant to topology formation. Bluetooth nodes are organized in small groups called *piconets*. Within every piconet there is a leading node called "master," and all other nodes are referred to as "slaves." A node may belong to multiple piconets, and we refer to such a node as a "bridge." A piconet can have at most 8 active members. Slaves do not directly communicate with each other but instead rely on the master as a transit node. Communication between nodes in different piconets relies on bridge nodes. A bridge node cannot be simultaneously active in multiple piconets. Bluetooth allows different activity states for nodes: active, sniffing, idle and parked. Data exchange takes place between two nodes only when both are active. Nodes periodically change their activity state.

Except for limiting the number of slaves in a piconet to 7, no other constraints exist regarding the assignment of roles to nodes. This flexibility, however, raises a number of questions. We briefly list below those that are most relevant to topology formation.

1. How should nodes select their role (master or slave)?

2. Which piconet(s) should a (slave) node join?

3. How many slaves should a master accept (below the specified maximum of seven)?

4. How many piconets should a bridge node belong to?

5. Should masters be allowed to be bridge nodes (slaves in other piconets)?

Specifically, consider questions 4 and 5. Because a bridge node can only be active in one piconet at a time, the greater the number of piconets to which a node belongs, the poorer the connectivity it can provide between them. This problem is compounded when the bridge node is a master in one piconet. This is because periods during which a master acts as a slave in another piconet correspond to a complete communication blackout for all the slaves of its own piconet. Thus, it is undesirable for a master to be a slave in other piconets, provided that such a slave/master role limitation does not introduce significant constraints when forming and modifying topologies. Note that this requirement was also considered in [19]. For the same reasons, we also assume that it is desirable for a bridge node to be involved in as small a number of piconets as possible, while preserving connectivity. This criterion is incorporated in some of the algorithms presented in the paper.

## 3. RELATED RESEARCH

In this section we mention very briefly a number of previous works that have also been motivated by the need to extend the standard Bluetooth specifications, if the technology is to be used for building ad-hoc networks. Those works span three major areas associated with ad-hoc networks: routing, resource management or scheduling, and topology formation. The latter being clearly the area of most relevance to this paper.

In [3] Bhagwat *et al.* present a source routing mechanism for Bluetooth scatternets, *i.e.*, networks formed from the interconnection of piconets. Johansson *et al.* [12] present a distributed scheduling policy for Bluetooth networks. The topology formation problem was first investigated in [19] by Salonidis *et al.* who presented a distributed topology construction scheme in Bluetooth networks. A basic assumption in the paper is that all nodes are within transmission range of each other, which does not often hold in many scenarios. The paper makes the very interesting observation that the average delay involved in synchronizing two nodes (the time spent in the inquiry and the page sequences before the nodes are able to exchange their clock information) is infinite if the nodes rely on a deterministic pattern of alternating between paging and paged modes. In [2], Zaruba *et al.* present "Bluetrees," a scatternet formation algorithm for cases in which the full reachability assumption does not hold. However, the Bluetree algorithm reduces the degree of the nodes by a series of rearrangements. There is no guarantee that these rearrangements actually terminate and, thus, the resulting topology may not satisfy the degree constraints. In [1] and [17], Basagni and Petrioli present two more algorithms, "Bluestars" and "BlueMesh". In [21], Tan *et al.* present "Tree Scatternet Formation," an online algorithm to build scatternets. However, it is unclear how the degree constraints and connectivity are satisfied, since only the root nodes of each fragment are allowed to merge different fragments. An interesting part of the paper is the model it proposes to evaluate the efficiency of a scatternet by approximating the average communication latency. In [13], Law and Siu also present a scatternet formation algorithm. The problem of topology formation was also the topic of [8], where we investigated the performance of a few simplistic topology formation algorithms from the standpoint of connectivity and convergence time. Finally, in [14], Marsan *et al.* address the problem of determining an optimal Bluetooth topology, based on an integer linear programming formulation derived from the Bluetooth-specific constraints. However, the complexity of the proposed algorithm is fairly high. Furthermore, their approach leads to a centralized optimization algorithm which raises the question of practical distributed implementation.

## 4. NETWORK MODEL AND PROBLEM COMPLEXITY

As a first step towards a systematic investigation of the connectivity issue, we formulate a mathematical model for the system objectives and constraints. Observe that there can be two types of communication links between any two nodes. One is a *physical* link, existing between any pair of nodes that are in communication range of each other. The other is a *logical* Bluetooth link, existing if the Bluetooth topology establishes a communication link between the two nodes. The physical topology graph is given (based on the positions and the transmission radii of the Bluetooth devices), while the logical topology graph is the output of the topology formation algorithm.

The logical topology graph must have certain properties. According to the Bluetooth specification, vertices assigned the role of a master can have a maximum degree[2] of 7. For vertices that will serve as slaves, it is desirable that their degree be kept as small as possible. Because a bridge node with a degree of 7 would represent a major bottleneck in the system, we assume that the degree constraint of 7 applies to the bridge (slave) nodes as well. Also, although not necessarily required, it is desirable that the graph be bipartite[3] with one set of vertices corresponding to masters and the other to slaves. Note that this ensures that no master assumes the role of bridge connecting two piconets.

Connectivity is then deemed feasible if there exists a connected[4] subgraph of the physical topology graph which satisfies the degree constraint (maximum degree of 7). The objective is to first detect whether connectivity is feasible. If so, then the aim is to construct a connected logical topology graph which satisfies the desired constraints. If connectivity is not feasible, then any logical topology graph will consist of "islands" or components[5]. In this case the objective is to minimize the number of components in the logical topology graph.

Note that a connected logical subgraph exists if and only if the physical topology graph has a spanning tree[6] that satisfies the degree constraints of a logical topology graph. This is because a spanning tree of any graph is connected. Also note that it is bipartite [10]. Let the degree of a spanning tree be the maximum degree of its vertices. The challenge is then to construct a spanning tree with degree less than or equal to 7, if one exists; deciding this is an NP-hard problem [7]. It follows that *deciding whether connectivity is feasible and constructing a connected logical topology graph which satisfies the desired*

[2]The degree of a vertex is the number of edges originating from the vertex.

[3]A bipartite graph is one where the vertex set can be partitioned in two sets such that there is no edge connecting the vertices in the same set.

[4]A graph is connected if there is a path between any two nodes.

[5]A component of a graph is a connected sub-graph which can not be expanded any further while retaining connectivity, *i.e.*, addition of a node in a component removes the connectivity.

[6]A spanning tree is a connected subgraph which does not have a cycle and spans all vertices in the graph.

*constraints is an NP-hard problem for a general physical topology.*

## 5. TOPOLOGY FORMATION ALGORITHMS FOR NODES WITH IDENTICAL POWER LEVELS IN A 2−DIMENSIONAL PLANE

In this section we approach the connectivity problem under two simplifying assumptions: All nodes lie on a two-dimensional plane and all nodes have the same transmission range. Under these assumptions, the time complexity of the connectivity problem becomes polynomial. The following lemma provides the cornerstone for designing a polynomial complexity, distributed, and dynamic algorithm which generates a connected logical topology whenever connectivity is feasible.

**Lemma 1** *Connectivity is feasible if and only if the physical topology graph is connected. A minimum weighted spanning tree (MST) in the physical topology graph, with the weight of an edge equaling the Euclidean distance between the nodes, is a connected logical topology graph which satisfies all the desired constraints.*

We first present the following result obtained by Monma *et al.* [16] which we will use in proving this lemma.

**Proposition 1** *Consider a complete*[7] *graph with nodes corresponding to points in a two-dimensional plane and the weight of the edges being the Euclidean distance between the corresponding vertices. Any minimum weighted spanning tree in such a graph has degree less than or equal to* 6.

**Proof of Lemma 1:** See [9].

Next, we consider the case when connectivity is not feasible, *i.e.*, when the physical topology graph is disconnected. The objective in this case is to construct a logical topology graph with the minimum number of components. The following lemma gives the basis for the procedure we will follow.

**Lemma 2** *The subgraph of the physical topology graph, consisting of the minimum weight spanning trees in each component, is a logical topology graph with the minimum number of components.*

**Proof of Lemma 2:** See [9].

*It follows from Lemmas 1 and 2 that constructing a minimum weighted spanning tree in the physical topology graph will provide a logical topology graph which (a) is connected if connectivity is feasible and (b) consists of the minimum number of components if connectivity is not feasible.* It is interesting to observe that a centralized minimum weight spanning tree algorithm has a complexity of only $O(E\log V)$ if the physical topology graph has $E$ links and $V$ nodes, whereas the construction of the logical topology graph is NP-hard in the general case, (*i.e.* without the assumptions made in this section).

If all nodes have low degrees, then the end-to-end path between certain nodes may be long and this may not be desirable for delay considerations. Thus, one may wish to have

[7]A graph is complete if it has edges between any pair of vertices.

somewhat larger piconets (desired piconet size can be a design parameter). This calls for algorithms which can tune the degree of masters to a desired value and the degree of bridges to a different, possibly lower value. In the next two sections we propose algorithms which can accommodate such a discriminatory treatment and, more importantly, are capable of generating connected topologies in cases in which the assumptions of this section do not hold, *i.e.*, higher dimensionality and relaxed power level assumptions.

## 6. TOPOLOGY FORMATION ALGORITHMS FOR NETWORKS WITH NODES IN 3−DIMENSIONAL SPACE

We focus on designing a topology where a node's degree does not exceed 7. Robins *et al.* [18] showed that the degree of a minimum weighted spanning tree can be as large as 13. Thus, unlike the 2−dimensional case, a MST-based algorithm is not guaranteed to satisfy the degree constraint of the masters. The problem needs, therefore, to be investigated in the framework of a minimum degree spanning tree, but, as discussed in Section 4, this is an instance of an NP-complete problem. We investigate heuristics and approximation algorithms for this purpose.

The MST algorithm does not give any analytical guarantee on the degrees of the nodes in the 3−dimensional case. In addition, it does not have the potential for separately controlling the degrees of the masters and the bridges (not even in the 2−dimensional case). We present next a topology design procedure based on an approximation algorithm guaranteed to generate a spanning tree with degree at most one more than the minimum possible value in any arbitrary graph [11]. More formally, let the degree of the spanning tree generated by this algorithm be $d$. Then any other spanning tree will have a maximum degree of $d-1$ or more. In the Bluetooth context this means that if $d \geq 9$, then any connected logical topology will have at least one node with degree greater than 7 and, therefore, connectivity will not be feasible. If $d \leq 7$, then connectivity is feasible and the spanning tree generated by this algorithm is a valid logical topology. If $d = 8$, then connectivity may or may not be feasible and any connected logical topology will have at least one piconet with 7 slaves. Thus, the "gray area" where this algorithm may fail and yet connectivity be feasible, is only for $d = 8$. We denote this algorithm as the "MDST" algorithm. The basic approach is to start with any spanning tree and replace edges from vertices of high degree with those from vertices of low degree. See[11] for a detailed description of the algorithm. MDST runs in polynomial time [11], $O\left(VE\log V\right)$[8], it "almost" always identifies whether connectivity is feasible, and if it is, MDST generates a connected logical topology.

Next, we discuss how to extend MDST to separately control the degrees of the masters and bridges. This algorithm reduces the maximum degree of nodes as much as possible, while our objective is now somewhat different. *The goal is to first satisfy a degree constraint of, say, p for all vertices (where p is the desired maximum number of slaves in a piconet), and subsequently preferentially reduce the maximum degree of the bridges down to a desired value (k).* Reducing the degree of all nodes uniformly need not attain this, since in most cases it results in both masters and bridges having degrees close to 2.

[8]More precisely, the run time is $O\left(VE\alpha(V,E)\log V\right)$, where $\alpha$ is the inverse of Ackermann's function, and grows slowly. For all practical purposes, $\alpha(V,E)$ can be treated as a constant.

| $N$ | 25 | | | | 50 | | | | 100 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $M_a$ | $M_m$ | $B_a$ | $B_m$ | $M_a$ | $M_m$ | $B_a$ | $B_m$ | $M_a$ | $M_m$ | $B_a$ | $B_m$ |
| MST | 2.2 | 3 | 2.3 | 3 | 2.3 | 3 | 2.1 | 3 | 2.4 | 4 | 2.3 | 3 |
| MDST | 1.9 | 2 | 2 | 2 | 2 | 2 | 2.1 | 3 | 2 | 2 | 2.1 | 3 |
| E-MDST | 5.9 | 7 | 2.7 | 3 | 6.1 | 7 | 2.4 | 3 | 6.1 | 7 | 2.7 | 3 |

Table 1
Evaluation of the algorithms in a 3-D clustered topology. $N$ stands for the number of nodes, $M_a$ for the average degree of the masters, $M_m$ for the max. degree of the masters, $B_a$ for the average degree of the bridges and $B_m$ for the max. degree of the bridges.

Our modification of MDST proceeds as follows. We start MDST with a spanning tree generated by BFS, which generates spanning trees of large degrees. MDST is then allowed to terminate when the maximum degree is reduced to $p$ (the desired piconet size). We denote this first minor modification of MDST as "M-MDST." It ensures that we do not end-up with "long" trees, but still does not allow for the separate tuning of the degree of master and bridge nodes. This is done through a second extension, which we call E-MDST. The algorithm starts with the spanning tree generated by M-MDST and proceeds to reduce the degrees of the bridges without increasing that of the masters beyond the degree constraint of $p$. The basic difference between MDST and E-MDST is that in E-MDST the edge replacement is used to decrease the degrees of bridges only, once the overall degree constraint of $p$ is satisfied by the M-MDST algorithm.

In order to evaluate the performance of the different algorithms we test their ability to generate connected topologies in two environments. The first consists of nodes uniformly distributed in a square of size 1 unit; the second consists of a "clustered topology" made of three square clusters of size 0.4. Next, a z-coordinate, uniformly distributed between 0 and 0.3 units, is assigned to each node. For each of these two types of node distributions, we evaluate the performance of the algorithms for different numbers of nodes $(25, 50, 100)$ and two different transmission radii (0.4 and 0.6 units), averaging the results over 100 runs. As our experiments have shown (Table 1), in all scenarios, node degrees remain well below the limit of 7. The average master's degree ($M_a$) indicates that E-MDST achieves its objective of generating a "bushier" topology, while at the same time attaining a small average bridge degree (around 2.7). These numbers seem to be the same in the 2-dimensional scenario, suggesting that dimensionality has no impact on these algorithms. See [9] for more detailed results on both the 2- and the 3-dimensional cases.

## 7. TOWARDS DISTRIBUTED AND DYNAMIC ALGORITHMS

In this section, we first illustrate how an MST based algorithm can be extended to operate in a distributed and dynamic setting. Because such an extension is not without complexity, we then introduce an algorithm that is distributed in nature, although it does not enjoy the same analytical performance guarantees as an MST algorithm.

## 7.1. Distributing an MST based algorithm

A minimum weighted spanning tree can be constructed by distributed computation at the nodes, *e.g.*, Prim's algorithm [5] for constructing minimum weight spanning trees can be distributized [6]. A node only needs to know an ordering of the weights of its incident edges. In the Bluetooth setting, a node acquires this knowledge while synchronizing with its neighbors. During this time, a node can measure the signal strength of the synchronization messages sent by its neighbors. If all nodes transmit these messages at the same power level, the signal will be stronger for a neighbor which is closer.

The same observation holds for addressing a dynamic scenario. For example, new nodes may join and existing nodes may leave the system. Nodes may be continuously on the move, and thus the neighbor set and the Euclidean distances between neighbors change. Thus the spanning tree needs to be updated in response to these topology alterations. There are efficient algorithms for dynamic update of spanning trees [4,20]).

However, the complexity of a distributed and dynamic version of the MST algorithm can be fairly high[9] (see [4,6] for details). This motivates the consideration of simpler distributed algorithms that rely on heuristics. In the next sub-section, we investigate a solution that is not guaranteed to generate a MST, and therefore enjoy any analytical guarantees, but may offer a favorable trade-off between performance and complexity.

## 7.2. A fully distributed and dynamic algorithm

The algorithm we investigate is based on the following local information based heuristic for selection of edges. Start with an empty logical topology. Consider two nodes A and B and the edge AB joining them. Draw a circle with A as its center and radius AB, and draw another circle with B as its center and radius BA. If there is no other node in the intersection of the two circles, then add the edge AB. If some node C lies in the intersection of the circles, then edge AB is not added

Note that a node C is in the intersection of the two circles if and only if its Euclidean distances to both A and B are smaller than AB. This can be determined from power measurements and information exchange during node synchronization. If such a node C is discovered after the edge AB has been added to the graph, then this edge can be dropped. Observe that this doesn't affect any other edge additions or deletions in the rest of the graph. Therefore, the decision of whether to add an edge or not is based solely on local information. Hence, there is no need to broadcast any information throughout the graph and there is no need to maintain edge or node states. This clearly reduces the number of exchanged messages as well as the complexity of this algorithm when compared to the distributed MST algorithm. This algorithm tries to approximate the MST, and in that context it is worth noting that the resulting graph will be a superset of the MST.

**Lemma 3** *The aforementioned heuristic generates a topology that is a superset of the Minimum Weight Spanning Tree (i.e., the topology generated by the MST algorithm).*

**Proof of Lemma 3:** See [9].

Since this heuristic may result in including more edges than in the MST, the resulting graph need not have a degree of 7 or less. But as we will see later, it typically contains

[9]The time complexity of a distributed MST algorithm is $O(V\log V)$ and the communication cost is $O(V\log V + E)$ messages [6].

| $N$ | $E$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $M_a$ | $B_a$ | $M/S$ | $D_{M/S}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 100 | 117.3 | 9.1 | 50.2 | 37.8 | 2.9 | 0.002 | 2.4 | 2.5 | 16.9 | 2.6 |
| 500 | 616.4 | 24.8 | 236.5 | 219.7 | 18.9 | 0.02 | 2.5 | 2.6 | 93.7 | 2.7 |
| 1000 | 1246.9 | 41.2 | 464.1 | 454.4 | 40.2 | 0.05 | 2.5 | 2.6 | 192.2 | 2.7 |

Table 2
Evaluation of the distributed and dynamic algorithm in a 3-D clustered topology. $N$ stands for the number of nodes and $E$ for the number of edges in the resulting topology. $D_i$ for the average number of nodes with degree equal to $i$, $M_a$ for the average degree of the masters, $B_a$ for the average degree of the bridges, $M/S$ for the number of nodes with a dual role and $D_{M/S}$ for their average degree.

only a few more edges than the MST and the resulting maximum node degree will most often not exceed 5. A more important issue is that because the resulting graph need not be bipartite, the algorithm might also lead to nodes having to assume the roles of both a master and of a slave. As discussed earlier, this is a situation that should be avoided if possible, even if nothing in the Bluetooth standard precludes it.

Again, we evaluated the algorithm in the same topologies we generated for MST, MDST and E-MDST (see Section 6). As our experiments have shown (see Table 2), in all scenarios, the degrees of the nodes are much below the limit of 7. The percentage of nodes that have to play a dual role (*i.e.* as both a master and a slave) is approximately between 17 and 19 percent of the total number of nodes, but their average degree is still low (around 2.7). Again, see [9] for more detailed results.

## 8. CONCLUSION

To summarize, this paper has presented a number of algorithmic results aimed at the problem of topology formation in Bluetooth ad-hoc networks. We have shown that the MST algorithm is the only one that is guaranteed to always satisfy Bluetooth degree constraints in a 2-D scenario. However, from a delay/throughput point of view it need not always be the case that minimal degrees for both masters and slaves is desirable. This motivated the introduction of the E-MDST algorithm, which allows for independent tuning of the degrees of masters and bridges, and therefore affords greater control on the resulting topology. Given the potentially high complexity of implementing distributed versions of those algorithms, we finally investigated a heuristic-based distributed algorithm that appears to satisfy the constraints of the Bluetooth technology.

The results of the work presented in this paper have provided the foundation for an actual design and implementation effort that we are currently pursuing. This effort is aimed at better assessing the implementation complexity of solutions based on a distributed MST algorithm and on the distributed algorithm of Section 7.2. It is being carried out by leveraging a detailed emulation of the Bluetooth stack, which allows us to precisely quantify the operation overhead of each algorithm in an operational Bluetooth environment. In addition we plan on exploring further the performance trade-off offered by the different algorithms studied in this paper.

## REFERENCES

1. S. Basagni and C. Petrioli. A scatternet formation protocol for ad-hoc networks of Bluetooth devices. In *IEEE Vehicular Technology Conference*, 2002.
2. S. Basagni, G. Zaruba, and I. Chlamtac. Bluetrees-scatternet formation to enable Bluetooth-based ad hoc networks. *ICC*, 2001.
3. P. Bhagwat and R. Seigal. A routing vector method (RVM) for routing in Bluetooth scatternets. In *MoMuC'99*, San Diego, CA, November 1999.
4. C. Cheng, I.A. Cimet, and S.P.R. Kumar. A protocol to maintain a minimum spanning tree in a dynamic topology. In *Proceedings of the ACM Symposium on Communications Architectures and Protocols*, Stanford, CA, 1988.
5. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.
6. R.G. Gallager, P.A. Humblet, and P.M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. on Programming Languages and Systems*, 5, 1983.
7. M. R. Gary and D. S. Johnson. *Computers and Intractability.* Freeman, 1979.
8. R. Guérin, E. Kim, and S. Sarkar. Bluetooth technology: Key challenges and initial research. *Conference on Network and Distributed Simulations*, 2002.
9. R. Guérin, S. Sarkar, and E. Vergetis. Forming connected topologies in Bluetooth adhoc networks. *University of Pennsylvania, Technical Report*, Available at http://m306pc7.seas.upenn.edu/mnlab/publications.html, 2002.
10. F. Harary. *Graph Theory.* Addison-Wesley, 1969.
11. D. Hochbaum. *Approximation Algorithms for NP-hard Problems.* PWS, 1995.
12. N. Johansson, U. Korner, and L. Tassiulas. A distributed scheduling algorithm for a Bluetooth scatternet. In *Proceedings of ITC'2001*, Salvador, Brazil, December 2001.
13. C. Law, A. K. Mehta, and K.-Y. Siu. Performance of a new Bluetooth scatternet formation protocol. In *Proceedings of MobiHoc'01*, Long Beach, CA, October 2001.
14. M.A. Marsan, C.F. Chiasserini, A. Nucci, G. Carello, and L. De Giovanni. Optimizing the topology of Bluetooth wireless personal area networks. In *Proceedings of INFOCOM'2002*, New York, NY, July 2002.
15. B. Miller and C. Bisdikian. *Bluetooth Revealed: The Insider's Guide to an Open Specification for Global Wireless Communications.* Prentice-Hall, 2000.
16. C. Monma and S. Suri. Transitions in geometric minimum spanning trees. *Discrete and Computational Geometry*, 8(3), 1992.
17. C. Petrioli and S. Basagni. Degree-constrained multihop scatternet formation for Bluetooth networks. In *IEEE Globecom'02*, Taipei, Taiwan, November 2002.
18. G. Robins and J. Salowe. On the maximum degree of minimum spanning trees. *Proceedings of ACM Symposium on Computational Geometry*, 1994.
19. T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Distributed topology construction of Bluetooth personal area networks. In *Proceedings of INFOCOM'01*, 2001.
20. K. Siu, P. Narvaez, and H. Tzeng. New dynamic algorithms for shortest path tree computation. *IEEE/ACM Transactions on Networking*, 8(6), 2000.
21. G. Tan, A. Miu, J. Guttag, and H. Balakrishnan. Forming scatternets from Bluetooth personal area networks. *MIT Technical Report*, MIT-LCS-TR-826, October 2001.