On the Complexity and Distributed Construction of Energy-Efficient Broadcast Trees in Wireless Ad Hoc Networks

Ashwinder S. Ahluwalia and Eytan H. Modiano, *Senior Member, IEEE*

***Abstract*—This paper addresses the energy-efficient broadcasting problem in ad hoc wireless networks. First, we show that finding the minimum-energy broadcast tree is NP-complete. We then develop a distributed clustering algorithm that computes energy-efficient broadcast trees in polynomial time. Our distributed algorithm computes all N possible broadcast trees simultaneously, while requiring $O(N^2)$ messages to be exchanged between nodes. We compare our algorithm's performance to the best-known centralized algorithm, and show that it constructs trees consuming, on average, only 18% more energy. We also consider the possibility of having multiple source nodes that can be used to broadcast the message and adapt our algorithm to compute energy-efficient broadcast trees with multiple source nodes. We observe a reduction in the amount of energy needed to form the broadcast tree that is linear in the number of source nodes.**

***Index Terms*—Broadcast, complexity, energy efficiency, wireless networks.**

I. Introduction

ENERGY-EFFICIENT communication is of paramount importance in networks of small battery-operated devices (e.g., sensor networks). In this paper, we are interested in the construction of energy-efficient broadcast trees in ad hoc networks, as posed in [8]. Starting with a given source node s, the problem is to find a broadcast tree that allows s to send a message to all other nodes, using the minimum amount of energy. Although it is tempting to do so, we do not simultaneously deal with other issues such as channel contention and mobility—we believe that a well-formed solution to this problem can serve as an intuitive start to algorithms that also include other issues.

We focus on a specific type of ad hoc network where all nodes are stationary (usually referred to as a "static" ad hoc network), and equipped with an omnidirectional transmitter. We assume that the transmission range of the transmitter can be adjusted from 0 up to a maximum range $R_{\max}$. Previous energy-efficient routing research has focused on environments where the transmitter can only be turned off or transmit at range $R_{\max}$ ([9]). In that case, the minimum-energy broadcast problem amounts to finding a broadcast tree with the minimum number of transmitting nodes. This problem is the connected dominating-set problem that is known to be NP-complete and for which, many heuristics exist [9].

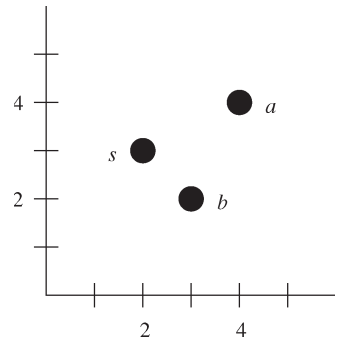


Fig. 1. Multicast advantage.

In contrast, our model assumes that transmitters can adjust their power levels so that they can operate at any range in between zero and some maximum range $R_{\max}$ ([6], [10]–[14]). The transmitter is referred to as omnidirectional because it does not focus the message transmission in a particular direction. Therefore, when the omnidirectional transmitter sends a message at range r, all nodes within distance r can receive the message, regardless of their position. Naturally, when the node transmits a message at a higher range, it consumes more power. In analyzing the range–power tradeoff, we adopt a common communications model, where the power required to transmit the message is proportional to r^α (typically $\alpha \geq 2$).

Note that because each node can only transmit up to distance $R_{\max}$, it is possible that the source node s cannot reach all nodes in the network directly. Therefore, some nodes will have the responsibility to forward messages on behalf of s. We can then rephrase the problem as one of assigning each node n_i a range r_i at which to forward received messages. The total cost of the broadcast tree can then be expressed as $\sum r_i^\alpha$.

This is a very uncommon cost function for graph-connectivity problems, in that cost is node weighted instead of edge weighted. Consider the example shown in Fig. 1. In this situation, we denote the cost incurred when node s transmits at a distance just large enough to reach b as $\mathrm{Power}(s, b)$, and define $\mathrm{Power}(s, a)$ analogously. If s attempts to send a message to node b, this message will not be received by node a because a's distance to s is larger than b's distance to s ($d_{as} > d_{bs}$). However, s could optionally transmit at $\mathrm{Power}(s, a)$, in which case the message would be received by both a and b (because $d_{bs} < d_{as}$). By transmitting to a, s can get a transmission to b

Manuscript received May 14, 2002; revised November 25, 2003; accepted July 31, 2004. The editor coordinating the review of this paper and approving it for publication is Y.-D. Yao.

The authors are with the Massachusetts Institute of Technology, Cambridge, MA 02139-4307 USA (e-mail: ash@mit.edu; modiano@mit.edu).

Digital Object Identifier 10.1109/TWC.2005.853824

for "free," due to the use of omnidirectional antennas. In [8], this was referred to as the "multicast advantage." In general, the power required for a node s to transmit a message to a set of nodes N is $\max_{n\epsilon N} \mathrm{Power}(s,n)$. A transmission at this power will be received by all nodes in N.

This paper is organized as follows. In Section II, we formulate the minimum-energy broadcast-tree problem and prove that it is NP-complete. In Section III, we consider distributed algorithms for the construction of energy-efficient broadcast trees and compare those to the centralized algorithm of [8]. In Section IV, we consider the case where the broadcasts can be initiated at multiple source nodes and adapt our algorithm to compute energy-efficient broadcast trees with multiple source nodes. Finally, in Section V, we conclude the paper with a discussion of our results, assumptions, and future directions.

II. Problem Formulation and Complexity

We can state the ad hoc wireless-broadcasting problem as follows. Given a set of nodes N and their coordinates on the two-dimensional plane. Additionally, we are given a range R, which represents the maximum distance any node can transmit a message, and a constant $\alpha > 0$, representing the power-loss exponent. We construct the undirected graph $G = (V, E)$, where $V = N$ and $(i,j)\ \epsilon\ E \iff d_{ij} \leq R$. Assuming this graph is connected, the wireless-broadcasting problem can be stated as follows:

> Given a source $s\ \epsilon\ N$ construct the minimum cost directed tree T (rooted at s) that connects s to every node in $N - \{s\}$ via a directed path according to the following cost function. Define $f(x) = \max\{d_{xj}^{\alpha} : (x,j)\ \epsilon\ T\}$. Then cost of T is defined as $\sum_{n\epsilon N} f(n)$.

Hereafter, we refer to the decision version of this problem, where we are asked to determine whether there exists such a tree with cost less than $l\ \epsilon\ \mathbb{Z}^{+}$, as BCAST. Naturally, one is interested in the complexity of the above problem. Unfortunately, the first contribution of this paper, stated in Theorem 1 below, tells us that this problem is computationally difficult and cannot be solved in polynomial time.

Theorem 1: BCAST is NP-complete.

Proof of this theorem is based on a reduction from the connected-node cover problem in planar graphs, and is given in the Appendix.

III. Distributed Broadcast-Tree Algorithms

Wieselthier, *et al.* considered the above broadcast problem [8] and proposed a centralized algorithm to construct energy-efficient broadcast trees. They showed that their algorithm, the broadcast incremental protocol (BIP), performed well as compared to minimum-spanning trees (MST) and shortest path trees. The BIP algorithm assumes the same model for power–range tradeoff that we assume in this paper (power proportional to r^{α}), so it is particularly relevant. Additionally, to our knowledge, BIP is the best known algorithm for this problem [15]. Consequently, we will use the performance of BIP as a measuring stick in judging our distributed algorithm. As discussed below, BIP assumes that each node does not have a range limitation ($R_{\max} = \infty$). This is an important distinction from our formulation, in which we assume that each node has some predefined range limit that is due to the transmitter's power limitation. However, BIP can be easily modified to account for the transmission-range limitation.

BIP is a greedy algorithm that mimics Prim's algorithm [17] for constructing MSTs. Throughout its execution, BIP maintains a set of nodes T that denote the tree made so far (initially, $T = \{s\}$ and the power of s is set to 0). At each step, BIP attempts to increase the power of a node $t\ \epsilon\ T$ to reach a node in $n\ \epsilon\ N - T$. Specifically, BIP increases the power of the node t that requires the least additional power to reach a node in $N - T$. The node n is then added to T, and the process is repeated until $T = N$. Once the tree is constructed, a sweep algorithm is run on the tree to reduce the power of nodes in specific cases where a node can be reached by multiple transmitters.

Although BIP has already been shown to construct low-cost trees, it's centralized nature requires one node to collect the position information of every node in the graph, compute the BIP tree, and distribute the solution to all other nodes in the network (alternatively, each node can gather the information and compute the tree independently). This can result in considerable time, message complexity, and power consumption. Additionally, this requires that the node performing the computation also has considerable resources (energy, processor, and memory). In the low-cost resource-limited environment that is typical in ad hoc networks, this may not always be feasible. These reasons motivate a need for a localized distributed algorithm that can compute broadcast trees efficiently. A localized algorithm is one in which nodes' decisions are based on network conditions within some limited distance [16].

In this section, we describe a localized distributed algorithm that computes broadcast trees. Our algorithm requires limited exchange of information between nodes that are within transmission range of each other. In the first portion of the proposed distributed algorithm, nodes calculate a clustering on the graph [9], [18]. Then, the clusters are joined together using a well known distributed algorithm for computing MSTs in directed graphs. A useful feature of our algorithm is that it constructs energy-efficient clusters. While our primary interest is in the formation of an energy-efficient broadcast tree; this clustering approach can be very useful for general (nonbroadcast) energy-efficient communications.

A. Distributed Construction of Broadcast Trees

At the beginning of the algorithm each node has the following information.

1) Each node i knows the distance to every node in i's neighborhood. A node's neighborhood is defined as the set of nodes that are within distance R (the maximum distance that a node can transmit a message). Nodes that are in i's neighborhood are referred to as neighbors of i.
2) Each node i also knows the distance of each neighbor to every node in the neighbor's neighborhood. We refer to the set of i's neighbors, and i's neighbors' neighbors as i's two-hop neighborhood.

As an example, this information could be gathered by determining pairwise node delay via timestamps. Notice that each node only requires localized information about some small portion of the network (the two-hop neighborhood). This is a key difference from previous algorithms that require each node to have global network information. Also, note that this is only meaningful in networks with limited range—if each node had unlimited range, having two-hop neighborhood information would be equivalent to having global information. In networks with limited range however, the two-hop neighborhood may constitute a small fraction of the graph. Of course, this also holds for networks in which the range of nodes is limited for reasons other than inherent transmitter limitations (e.g., interference avoidance).

In our initial development of a distributed algorithm, we assume that the network is synchronized via a global clock, no messages are lost, and that there is no interference. We then extend the algorithm to work without the benefit of a global clock, in networks where interference and packet loss is possible.

1) The Formation of Clusters: In the first phase of the algorithm, a clustering is constructed on the nodes using the aforementioned distance information. Once this phase is complete, each node will be assigned to at least one cluster, and each cluster will have one "clusterhead" node. We define the cost of a particular cluster as the power required for the clusterhead node to transmit to all other nodes in the cluster (in one transmission). In addition, we consider the cost of a particular clustering to be the sum of the costs of its clusters. Given this cost function, we attempt to develop a minimum cost clustering.

a) Centralized clustering algorithm: Before describing our distributed clustering algorithm, we first describe a centralized clustering scheme. Throughout the execution of the centralized algorithm, each node i's range is referred to as r_i, and each node is either unmarked or marked, reflecting its membership in a cluster. The algorithm begins with $r_i = 0$, for all i, and all nodes unmarked, and proceeds as follows.

1) For each node i, compute the function $\alpha_i(r)$. If i was to increase its range to r, this function represents the average additional energy cost incurred for each new unmarked node within distance r of i. More precisely,

$$\alpha_i(r) = \frac{P(r) - P(r_i)}{U_i(r_i, r)}$$

$$r_i < r \leq R$$

$P(x)$ power to transmit at range x;
$U_i(x_1, x_2)$ number of unmarked nodes in between distances x_1 and x_2 of i;
r_i node i's present transmission range.

2) For each node i, compute the range at which $\alpha_i(r)$ is minimized. This is the most cost-efficient range increase (in a greedy sense) for node i. Denote this range as rmin_i, and the value of α at this range as amin_i.
3) Find the node j that has the smallest value of amin—this is the node that (globally) has the most cost-efficient range increase. Increase r_j to rmin_j, and mark nodes j and all nodes within distance rmin_j of j.
4) Repeat steps 1)–3) until all nodes are marked.

Once the above algorithm terminates, the final r_i values specify a clustering (because nodes are only marked if they belong to a cluster). Each node with nonzero r_i is considered a clusterhead, and all nodes within distance r_i of i are considered members of i's cluster (note that a particular node may be a member of more than one cluster). Since we choose the range increase that minimizes the average additional cost incurred per node marked, we are hopeful that the clustering produced is cost efficient.

Interestingly, the greedy behavior of the above algorithm can be implemented distributively, with one important difference. The distributed algorithm, for reasons stated earlier, does not attempt to find a global minimum, but instead attempts to find the local minima. Although this may result in less power-efficient clusterings, such inefficiencies are inherent to many localized algorithms.

b) Distributed clustering algorithm: As in the global algorithm, each node i maintains a range value r_i initially set to 0, and is initially unmarked. Additionally, we ensure that each node i maintains up-to-date values of: 1) r_j for all neighbors j; and 2) Marked status of all nodes in the two-hop neighborhood. The algorithm we propose operates in stages. During each stage, local minima are computed, and the ranges of some nodes are consequently increased. Information about the range increases are then propagated. Once this has been completed, each node has updated its state information to reflect the last stage's changes, and the next stage begins.

Note that because we are assuming that all nodes are synchronized, we describe the algorithm in stages and substages, where each stage and substage begin on predefined clock boundaries. In each stage, each node executes the following substages:

Substage 1) If node i is unmarked, it computes, for each neighbor j, the minimum value of $\alpha_j(r)$ for $r \geq \mathrm{distance}(i, j)$. That is, i finds the most cost-efficient range increase for j, looking only at those ranges that would allow i to be a member of j's cluster. Denote the value of the range and α found through this computation as $\mathrm{rmin}_{j\to i}$ and $\mathrm{amin}_{j\to i}$, respectively.
Each node i then finds the neighbor node k with minimum value of $\mathrm{amin}_{k\to i}$ and sends k a Preferred message containing range value $\mathrm{rmin}_{k\to i}$. This message is sent at full power, so that it can be heard by all of i's neighbors.
If i is already marked, it does not participate in this substage.

In addition, each node i (marked or unmarked) executes the following steps.

Substage 2) At this substage, i has received all Preferred messages from its unmarked neighbors. If i receives a Preferred message with range value r' from all unmarked nodes within distance r' (indicating that it is a local minima), i increases

r_i to the value r'. Upon increase, it transmits a Range_Increase message at maximum power, telling all neighbor nodes that i has increased its range to r'. If i is not already a member of a cluster, it also broadcasts a Marked_Status message to its two-hop neighborhood, indicating that i has been marked (by virtue of becoming a clusterhead).

Substage 3) If i receives a Range_Increase message from a neighbor j, such that the distance from i to j is less than the new value of r_j, i is a member of j's cluster. Consequently, if i is not already a member of another cluster, it broadcasts a Marked_Status message to its two-hop neighborhood, indicating that i has been newly marked. Additionally, at this substage, i may receive a Marked_Status message from a neighbor that has just become a clusterhead (in the previous substage). It forwards this message at maximum power (to ensure that it goes to all nodes two hops away from the clusterhead).

Substage 4) At this substage, each node i may receive a Marked_Status message from a newly marked neighbor j. It retransmits this message at maximum power, to ensure that it reaches j's two-hop neighborhood. At the next substage, all messages will have reached their intended receivers. Therefore, in the next substage, each node i will have up-to-date information on r_j for all neighbors j, and the Marked status of every node in the two-hop neighborhood. After this substage, a new stage begins.

The algorithm terminates once all nodes have been marked (and hence no Preferred messages are being generated). Because nodes only mark themselves when they have become a member of a cluster, this also means that, upon termination, the final values of r_i produce a clustering. Note the following properties of this algorithm.

1) The algorithm terminates in a linear number of stages. Consider any stage of the algorithm where not all nodes have been marked. We show that at least one new node will be marked in this stage. Let amin_i be the minimum value of α_i for node i. There must exist some node j for which amin_j is minimum over all nodes. Because this is the global minimum, in the next stage, all unmarked nodes within distance rmin_j of j will send j a Preferred message with range value rmin_j (if not, amin_j would not have been the global minimum). Therefore, in the next stage, j will increase its range, and some set of previously unmarked nodes will be newly marked. This further implies that in every stage, at least one node is marked, completing the proof.
2) The algorithm uses $O(N^2)$ messages. In each stage, there is at most one Preferred message and one Range_Increase message per unmarked node, resulting in $O(N)$ messages per node per stage, and $O(N^2)$ messages total. Additionally, each node transmits a Marked_Status message when it has been marked (this occurs once per node throughout the algorithm). Because each node has two-hop neighborhood information, it can compute a spanning tree upon which to forward this Marked_Status message. Therefore, it takes at most $O(N)$ messages to forward the Marked_Status message to the two-hop neighborhood. Hence, we have at most $O(N^2)$ Marked_Status messages, and $O(N^2)$ Preferred/Range_Increase messages.

A clustering sweep procedure: Note that in the clustering produced by the algorithm above, it is quite possible that a node is simultaneously a member of more than one cluster. That is, clusters may overlap. As in the BIP procedure, there is a similar opportunity to implement a "sweep"-like algorithm [8]. This "sweep" goes through the clustering in a distributed manner, and finds nodes whose range can be reduced, while still making sure every node is still a member of at least one cluster. The ranges of these nodes are then reduced to produce a lower power clustering. We implemented a very simple cluster-sweep procedure that performs this operation.

2) Joining Clusters Together: After a clustering has been found, we use a well-known distributed algorithm for constructing directed MST (DMST) [7] to join the clusters together. Specifically, this can be accomplished as follows.

1) Construct the directed graph $G' = (V', E')$ where $V' = V$, and the cost of each edge (i', j') is equal to $\max(0, P(\mathrm{distance}(i', j')) - P(r_i))$, where $r_i < R$, and $P(x)$ denotes the power to transmit at a range x. This represents the incremental power required to establish a link from i' to j', after the clustering has been performed.
2) Once the cost of each edge has been computed, we run the algorithm for computing a DMST on G' for source s. By definition of the directed spanning tree, we will have constructed a broadcast tree rooted at s.

Our distributed algorithm first computes a clustering, sweeps the clustering, and then runs the DMST algorithm to join the clusters together. Note that the algorithm in [7] computes the DMST rooted at every node with $O(N^2)$ message complexity. Therefore, our algorithm computes the broadcast tree rooted at every node simultaneously.

3) Implementation Considerations: Although this synchronous algorithm works fine when there is a global clock and we assume no messages are lost or reordered, this is not at all a reasonable expectation of real-world environments. In practice, keeping global clocks up to date requires considerable message complexity and node coordination. Additionally, messages can be lost in wireless communication, requiring retransmissions that cause arbitrary message delays. Although the presentation in [7] demonstrated that clusters can still be joined successfully under these conditions, the clustering phase of our algorithm is of concern.

To extend our clustering algorithm to work without global synchronization, and in the presence of arbitrary message delays we can note the following. In each stage, three substages occur: 1′) Preferred messages are sent, 2′) some nodes increase their range and send a Range_Increase, and 3′) Marked states are propagated. If we ensure a node does not proceed to substage $k + 1'$ without receiving the messages of substage k',

the clustering developed will be identical to the synchronous case. For a node to make the same decision as in the synchronous algorithm in a substage $k+1'$, it must have received all messages destined to it, and sent off in any previous substage (all state information must be up to date). Hence, the algorithm can tolerate reordering of messages within a substage as long as the algorithm requires nodes to wait for all messages to arrive before proceeding to the next substage.

We can do this as follows. Assume that every node has up-to-date information from the previous stage (values of r_i for each neighbor, and the Marked status of each node in the two-hop neighborhood). In substage $1'$, each unmarked node sends a Preferred message, as in the synchronous algorithm. Each node is not allowed to enter a substage $2'$ until it has received all Preferred messages from its unmarked neighbors. Once a node is allowed to enter substage $2'$, it sends off a Range_Increase message as in the synchronous algorithm. If it will not increase its range, the node sends off a Non_Action_Range_Increase message to indicate this. As with substage $2'$, no node is allowed to enter substage $3'$ until a substage $2'$ message is received from all neighbor nodes. Once a node enters substage $3'$, it sends off a Status message only if it was unmarked in the last stage. Once a node has received a Status message from every unmarked neighbor, it composes and sends a Status_Summary message, which lists all nodes that have been marked in this stage (this ensures that a Marked update from a node travels two hops). A node is then not allowed to enter the next stage until a Status_Summary message has been received from all neighbors (note that once this is done, the node has received all messages destined for it in this stage, and therefore has up-to-date information). This algorithm ensures that no two neighbor nodes are out of synchronization by more than one substage, regardless of message delay.

Since the above algorithm tolerates messages delivered with arbitrary delay, it can further be extended to tolerate networks where messages can be lost through the use of a link-layer retransmission protocol (ARQ). Such a protocol guarantees the eventual delivery of packets, although the delivery time of the packet may vary based on the need for retransmission. However, since the above algorithm can tolerate packets that are arbitrarily delayed, we are assured that it will terminate successfully.

B. Simulation Results

To gauge the performance of our algorithm against BIP, we simulated the performance of BIP, and our distributed algorithm, in networks restricted to the 1 by 1 unit square. In simulating these algorithms at a particular network size, we first constructed a set of 100 instances, each having the same number of nodes. For each instance, nodes were randomly placed (with uniformly distributed coordinates)in the unit square, and one node was randomly chosen to be the source. Each algorithm was then executed on each of the 100 instances. After a tree was computed, the appropriate sweep procedure was executed on the tree. We compared the performance of our algorithm to BIP for networks with between 10 and 300 nodes. The results (shown in Fig. 2) display the relative performance of our

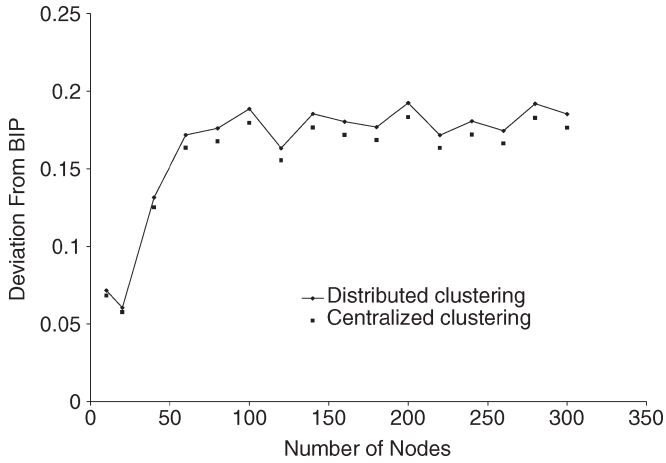


Fig. 2. Distributed versus centralized cluster formation.

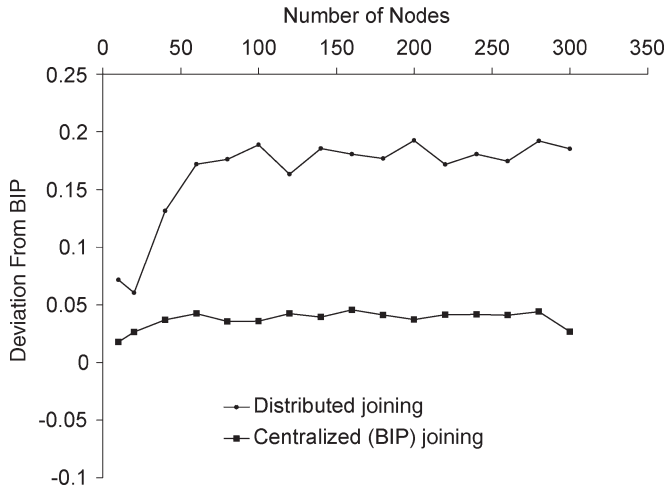


Fig. 3. Distributed versus centralized cluster joining.

algorithm as compared to BIP, averaged over the 100 instances. As can be seen from the figure, our distributed algorithm consumed, on average, 18% more energy than the centralized BIP.

Also shown in the figure is the result of using the centralized clustering algorithm described earlier.[1] As can be seen from the figure, the centralized clustering algorithm performs only marginally better than the distributed clustering algorithm. This tells us that the inefficiency due to the distributed formation of clusters is minimal. In order to further explore this issue, we also compared our algorithm to an alternative algorithm that uses a variation of BIP to join the clusters. This algorithm uses our distributed clustering procedure to form the clusters, and then uses BIP to join the clusters, as opposed to the distributed spanning-tree algorithms (DMST). The results of this new algorithm are shown in Fig. 3. As can be seen from the figure, joining the clusters together using the centralized BIP leads to a significant reduction in total energy consumption, and performs nearly as well as the centralized BIP. This, taken with the results of Fig. 2, suggest that the inefficiency in the

[1]Clusters were formed using the centralized algorithm described earlier; however, they were joined using the distributed spanning-tree algorithm (DMST).

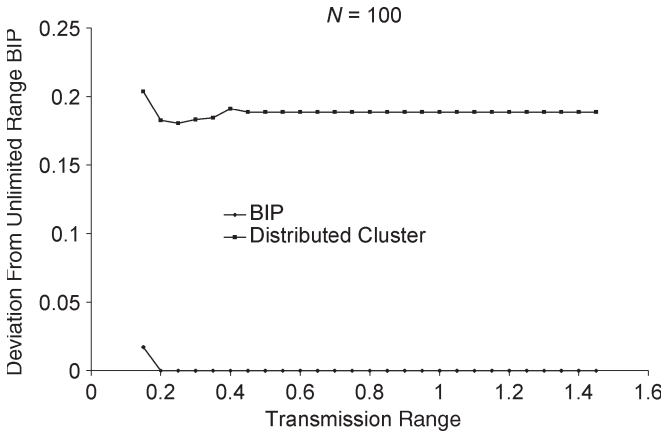


Fig. 4. Impact of transmission range ($N = 100$ nodes).

distributed construction of the broadcast trees is not in the distributed formation of the clusters, but rather in the distributed "joining" of clusters.

It is interesting to note that our results were largely insensitive to the maximum transmission range of the nodes. To explore this, we considered the implementation of both BIP and our distributed algorithm when the maximum transmission range of the nodes is limited. This allows for a direct and fair comparison between BIP and our distributed algorithm. In Fig. 4, we plot the deviation of our algorithm from BIP versus the maximum transmission range of the nodes. We also plot the deviation of BIP with limited transmission range from BIP with unlimited range. As can be seen from the figure, we found that as long as the maximum transmission range of the nodes was greater than the minimum range required to achieve connectivity, the cost of forming a broadcast tree using either BIP or our distributed algorithm was largely unchanged. Notice that in the case of BIP, the energy consumption with a limited transmission range has a zero deviation compared to BIP with an unlimited range. Hence, at all relevant transmission ranges, our algorithm consumed about 18% more energy than BIP. Moreover, this observation implies that most nodes in the broadcast tree use relatively small transmission ranges.

IV. Multiple-Source Broadcast

In our analysis so far, we have assumed there is only one source from which one message is sent to all other nodes in the network. However, in many networks, it may be useful to have several sources send the same message to every node in the network. For example, in a sensor network, certain special "supervisor" sensors may have satellite links to a central communication center. If a control command is to be issued from the communication center, destined to reach all sensors, it can be sent to all supervisor nodes via satellite, and then forwarded to the entire sensor network. In such a scenario, significant energy could be saved by broadcasting the message from many supervisor nodes, instead of just one. We investigate this possibility, and construct variants of our distributed algorithm and BIP to work in this case.

The multiple-source problem can be formally presented as follows. We are given a set of nodes N and their coordinates on the two-dimensional plane. Additionally, we are given a range R, which represents the maximum distance any node can transmit a message, and a constant $\alpha > 0$, representing the power-loss exponent. We construct the undirected graph $G = (V, E)$ where $V = N$ and $(i, j) \epsilon E \iff d_{ij} \leq R$. Assuming this graph is connected, the problem can be stated formally as:

> Given a set of sources $S \subseteq N$ construct the minimum cost directed forest[2] F (where each tree in this forest is rooted at some $s \epsilon S$) such that, $\forall n \epsilon N - S$, $\exists s \epsilon S$ such that there is a directed path from s to n in F. The cost of F is defined as $\sum_{n \epsilon N} f(n)$, where $f(x) = \max\{d_{xj}^{\alpha} : (x, j) \epsilon F\}$.

Note that because this is a generalization of BCAST, the decision version of this problem is also NP-complete.

1) Multiple-Source BIP: To modify BIP to work for multiple sources, we can note the following: the proof of BIP's correctness relies on the fact that at each iteration, the tree being constructed T contains a path from the single source s to every node in T. Each time a node is added to T, this property is maintained. Therefore, the tree at the end of the algorithm contains a path from the source to every other node (and for this reason, is a valid tree). Analogously, in the multiple-source problem, we can maintain a forest F such that there is a path to every node in the forest from at least one source node. When adding a node from $N - F$ to F we can use the same criteria as in the original BIP (least additional power). If we continue to add nodes in this way, the forest constructed at the end of the algorithm will ensure a path to every node from some source node. This algorithm can be described more completely as follows.

Throughout its execution, maintain a set of nodes F that denote the forest made so far. Additionally, maintain a power p_i for each node in F (initially, $F = S$ and p_s is set to zero for each node $s \epsilon S$). At each step, increase the power of a node $f \epsilon F$ to reach a node $n \epsilon N - F$. Specifically, increase the power of the node f that requires the least additional power to reach a node in $N - F$ (the additional power for f to reach node n is $\text{Power}(f, n) - p_f$). Once a node n has been chosen to be added by some node f, n is added to F with $p_n = 0$, and p_f is increased to $\text{Power}(f, n)$. This process continues until $F = N$.

We consider the above algorithm as an extension of BIP to the multiple-source problem.

2) Distributed Algorithm With Multiple Sources: To understand how we might extend the distributed algorithm, we must generalize each step that it takes. This would include the clustering algorithm, the clustering sweep, and the DMST algorithm used to join clusters together. Note that the distributed clustering algorithm presented above generates a clustering that is source independent. That is, the clustering developed is not effected by the choice of source node. Therefore, this algorithm can still be used in the multiple-source case. A similar argument can be applied to the clustering sweep.

Peculiarly, note also that if we did not modify the DMST algorithm we would still construct a valid tree. However, this

[2] A forest is a collection of subtrees.

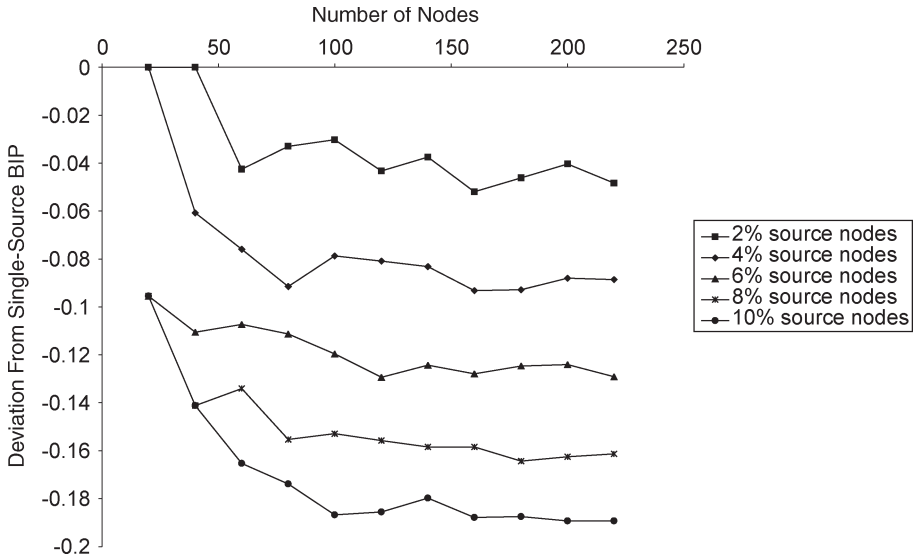


Fig. 5. Performance of the multiple-source variant of BIP, relative to BIP with one source.

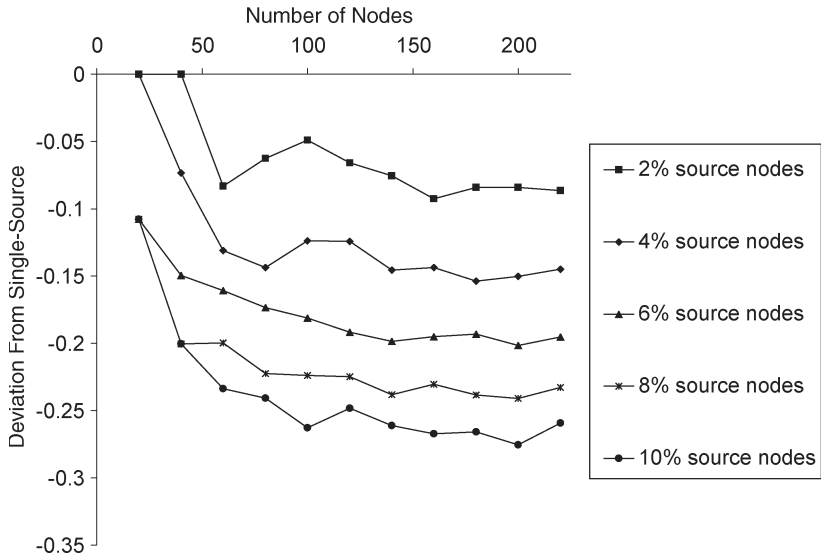


Fig. 6. Performance of the multiple-source variant of our distributed algorithm, relative to algorithm with one source.

tree would only utilize one source, and may consequently miss an opportunity for cost savings. To fix this, we first look at how to remedy the multiple-source problem in graphs with nonnegative edge weights (with no geometric restriction), where the cost of a forest is the sum of the edge weights. In this environment, we add a directed edge to the original graph between each pair of sources in S with cost zero (if the edge already exists, we modify its cost to be zero). Note, that if we now compute the minimum-cost single-source directed spanning tree rooted at any node s in S, the resulting tree will specify the minimum-cost forest.

In the multiple-source BCAST environment, where the cost function and geometric restrictions are different, we employ a similar idea. We modify the use of the DMST algorithm as follows. Note that originally the cost of each directed edge (i', j') in our graph G' is set to $\max(0, P(\text{distance}(i', j')) - P(r_i))$, where r_i is the range assigned from the clustering-sweep algorithm, and $P(x)$ denotes the power to transmit at a range x. We add a directed edge to G' between each pair of sources in S with cost zero. We then run the DMST algorithm of [7] on this modified graph. Note that although the DMST algorithm calculates the broadcast tree rooted at every node, our modified algorithm is based on a DMST algorithm that computes a forest with only source nodes as roots.

3) Performance: We simulated the efficiency of the modified BIP algorithm and our distributed algorithm on graphs with multiple sources to gauge their performance. The results of our simulation are shown in Figs. 5 and 6. As discussed previously, for simulation at a particular network size, we first constructed a set of 100 instances, each having the same number of nodes. For each instance, nodes were randomly placed with uniformly probability in the unit square, and a random set of sources was chosen. Each algorithm was then run on each of the 100 instances. After a tree was computed from any algorithm, the sweep procedure from [8] was run on the tree before comparison. The results were then averaged over the 100 instances.

Fig. 5 demonstrates that the power of the BIP tree decreases at about twice the rate of increase in the number of source nodes. Hence, with 2% of the nodes being designated to be source nodes, a 4% reduction in power is achieved. Additionally, the cost savings begin to diminish as the percentage of source nodes increases. The behavior of our distributed algorithm, shown in Fig. 6, seems to be very similar. In this case, a reduction of 5% in energy is achieved with the addition of 2% in source nodes. Again, this reduction in energy use diminishes as the percentage of source nodes is increased. This seems to indicate that the use of additional sources is only marginally useful for the purposes of minimizing energy consumption. However, notice that the location of the source nodes was assumed to be random. It is rather likely that if source nodes were chosen optimally, the energy savings would be far more significant.

V. Conclusion

A primary contribution of this paper was in showing that the problem of forming minimum energy broadcast trees is NP-complete. Additionally, we developed a distributed algorithm that computes suboptimal broadcast trees using $O(N^2)$ message complexity in polynomial time. This algorithm computes all N possible broadcast trees (one for each on N possible source nodes) and only consumes 18% more power on average than trees produced by the centralized broadcast incremental protocol (BIP) algorithm. A nice feature of the algorithm is that it forms energy-efficient clusters that can be used for general energy-efficient communication (e.g., point to point). We also introduced the multiple-source broadcasting problem, where a number of nodes can be used as source nodes to broadcast the message. This idea of multiple source nodes opens an interesting avenue for future research. For example, the problem of optimal positioning of the source nodes, in order to minimize the amount of energy needed to form a broadcast tree, is an interesting direction for future research.

Another interesting direction for future research is to consider the effect of mobility. The distributed algorithms described in this paper would work effectively in static topologies or topologies with limited mobility. However, with significant mobility, the problem of finding and maintaining routes, even without consideration of energy, is challenging [1]. The distributed clustering algorithm described in this paper holds some promise of performing well in a mobile environment as the "clustering" is localized and, therefore, not as affected by mobility.

Appendix
Proof of NP-Completeness

Before presenting the proof of Theorem 1, we start by going over the theorems, algorithms and definitions that we use to prove BCAST's NP-completeness.

Node Cover: Given an undirected graph $G = (V, E)$, a node cover is a set of nodes $S \subseteq V$, such that for every edge $(i, j) \epsilon E$, $i \epsilon S$ or $j \epsilon S$.

Connected Node Cover: A connected node cover of a graph $G = (V, E)$ is a node cover S, such that the graph induced by S on G is connected.

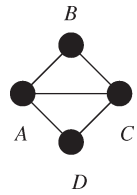


Fig. 7. POGD for the graph G with additional edge $\{B, D\}$.

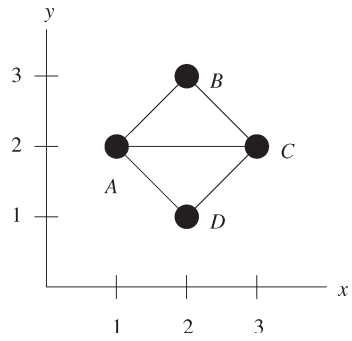


Fig. 8. G is also a planar graph. This demonstrates a planar embedding for the graph G.

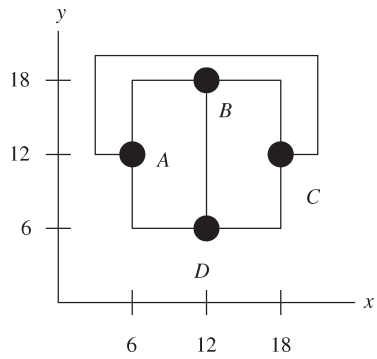


Fig. 9. POGD for the graph G.

Connected Dominating Set: A dominating set of a graph $G = (V, E)$ is a subset $S \subseteq V$, such that every node in V is either in S or is a neighbor of a member of S. A connected dominating set is a set S, such that the subgraph induced by S is connected, and S is a dominating set (Fig. 7).

Planar Graph: A planar graph $G = (V, E)$ is a graph that can be drawn in the plane without any edges overlapping. In other words, there exists a function $\pi_1 : V \rightarrow \mathbb{R} \times \mathbb{R}$, such that if we draw a point at $\pi_1(v)$ for all $v \epsilon V$, and then draw a straight-line segment from $\pi_1(i)$ to $\pi_1(j)$ in the plane for all $(i, j) \epsilon E$, no line segments will cross. The function π_1 is referred to as a planar embedding of the planar graph G (Fig. 8).

Planar Orthogonal-Grid Drawing: Given a planar graph $G = (V, E)$, a planar orthogonal-grid drawing (POGD) of G is a drawing on a grid such that each vertex is mapped to a grid point via some function $\pi_2 : V \rightarrow \mathbb{Z} \times \mathbb{Z}$, and each edge is mapped to a sequence of horizontal and vertical grid segments, such that no two edges ever cross (Fig. 9).

Unit Disk Graph: A graph $G = (V, E)$ is considered a unit disk graph if there exists a mapping $\pi_3 : V \rightarrow \mathbb{Q} \times \mathbb{Q}$ to points on the two-dimensional grid such that $(i, j) \epsilon E$ $\pi_3(i)$ and $\pi_3(j)$ are less than distance 1 apart (Fig. 10).

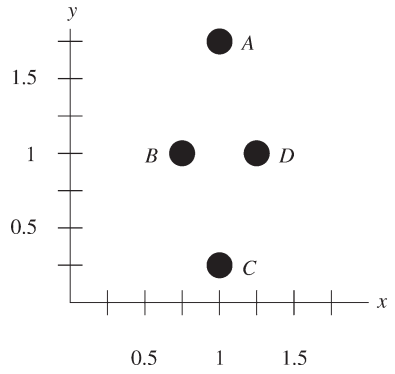


Fig. 10. Example showing G is also a unit disk graph. This is a demonstration of π_3.

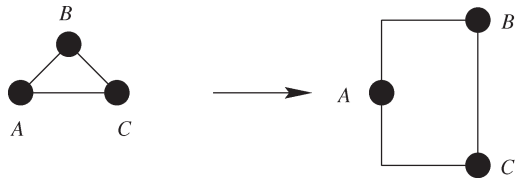


Fig. 11. Constructing the POGD in Step 1 of the reduction.

Having defined the terms above, we are now in a position to present the results of other papers used in our proof.

Theorem 2—NP Completeness of Planar Connected Node Cover: Given a planar graph $G = (V, E)$ of maximum degree less than or equal to 4, determining the existence of a connected node cover $V^* \subseteq V$ of G such that $|V^*| \leq k$, for some given $k \epsilon \mathbb{Z}^+$, is NP-complete. Proved in [3]. Hereafter, we refer to this decision problem as PLANAR.

Theorem 3—Orthogonal-Grid Drawings of Planar Graphs: Given a planar graph $G = (V, E)$ of maximum degree less than or equal to 4, an orthogonal-grid drawing of this graph can be drawn in polynomial time, such that the size of the grid is polynomial in $|V|$. Proved in [4].

Theorem 4—Connected Domination in Unit Disk Graphs: Finding a minimum-sized connected dominating set of a unit disk graph is NP-complete. We refer to the decision version of the connected dominating-set problem (i.e., "Does there exist a connected dominating set of size no more than k?") as CDSUDG. We reproduce the reduction used in the proof of this theorem (from [5]) below.

Given an instance of PLANAR, with graph $G = (V, E)$, maximum node-cover size $k \epsilon \mathbb{Z}^+$, we convert it to an instance of the CDSUDG problem as follows.

1) We first construct the POGD of G using the algorithm mentioned in Theorem 3 (this is done on an example graph in Fig. 11).
2) We then multiply the size of the grid by 6 so that each line segment of length one is mapped to a segment of length 6. At this point, we set $r = 1$ grid length (hereafter, we also refer to a node transmitting at range r as using one unit of power). This illustrated in Fig. 12.
3) Place a node at every grid point in the POGD, and denote this set of nodes as P [For example, if the line segment from (0, 0) to (0, 2) is in the orthogonal-grid drawing, P contains nodes at positions (0, 0), (0, 1), and (0, 2)]. Note that each vertex $v \epsilon V$ in the instance of PLANAR maps to a node in $p \epsilon P$, such that $\pi_2(v) = p$ (where π_2 is the function in the definition of a POGD). For each $p \epsilon P$ such that $\pi_2(v) = p$ for some $v \epsilon V$, we refer to p and all nodes in P that are within one grid length of p as the node region of v. Those nodes that are in the node region by virtue of being within one grid length of p are called the end nodes of that node region. The other node (the one that is mapped to from V via π_2) is referred to as the center node of this node region. See Fig. 13.
4) We then construct the set P_l as follows. Construct the subset $P' \subset P$, which contains all nodes in P that are not in node regions. Also, for future reference, we denote the set $P'' \subset P$ as the set of nodes in P that are not center nodes. P_l is then constructed such that: 1) each P_l node is placed at a grid point; 2) for each node in P', there is exactly one node in P_l located one grid length away; 3) for each node in P_l, there is exactly one node in P' located one grid length away; and 4) no node in P_l is within one grid length of any node in $P - P'$ (Fig. 14). This operation effectively creates a "layer" of nodes around the original POGD's edges, which is why we use the subscript l).

To complete the reduction, we construct a unit disk graph, so that every node in $P \cup P_l$ corresponds to a node in the unit disk graph, and edge (i, j) exists in the unit disk graph iff i and j's corresponding nodes are within distance 1 of each other.

Denote $|V|$ as the total number of nodes in the original PLANAR instance, and $|E|$ as the total number of edges. In the last step of the reduction in [5], the following lemma was proved.

Lemma 1: There is a vertex cover of size no more than k in the original PLANAR instance iff there is a connected dominating set in the corresponding unit disk graph of size no more than $|V| - |E| - 1 + k + |P''|$.

A. Proof of Theorem 1

We construct a reduction from PLANAR to BCAST inspired by the reduction used in [5], showing that we can convert any instance of PLANAR into an appropriate instance of BCAST in polynomial time. We confirm the correctness of our transformation by showing that every positive instance of PLANAR maps to a positive instance of BCAST, and that every negative instance of PLANAR maps to a negative instance of BCAST. This demonstrates that BCAST is NP-hard. We go on to prove that it is NP-complete, by showing $\mathrm{BCAST} \epsilon \mathrm{NP}$.

1) The Reduction: In proving the NP-hardness of BCAST, we can extend the reduction in [5] and use some of the properties derived there to prove the correctness of our reduction.

To extend the reduction in [5], we construct the BCAST instance from PLANAR instance as follows. First, we perform the reduction in [5] to an instance of CDSUDG. We then modify this reduction as follows.

Choose an arbitrary magnified POGD edge segment, such that one end of the segment corresponds to a center-node position (note that the POGD is magnified six times, so it must be six grid units long, and contain six nodes). Denote the first

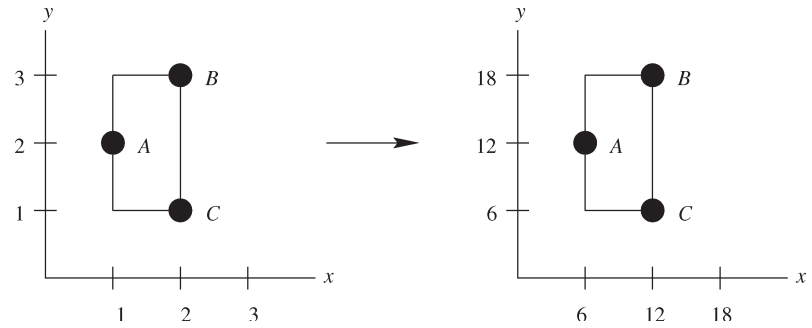


Fig. 12. Multiplying the grid size in Step 2 of the reduction.

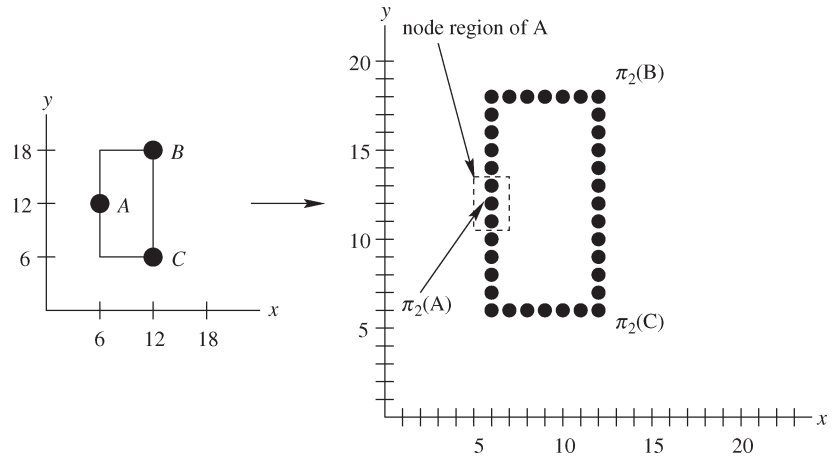


Fig. 13. Step 3 of the reduction. P is the set of nodes in the graph on the right. The end nodes in A's node region are at (6,11) and (6,13).

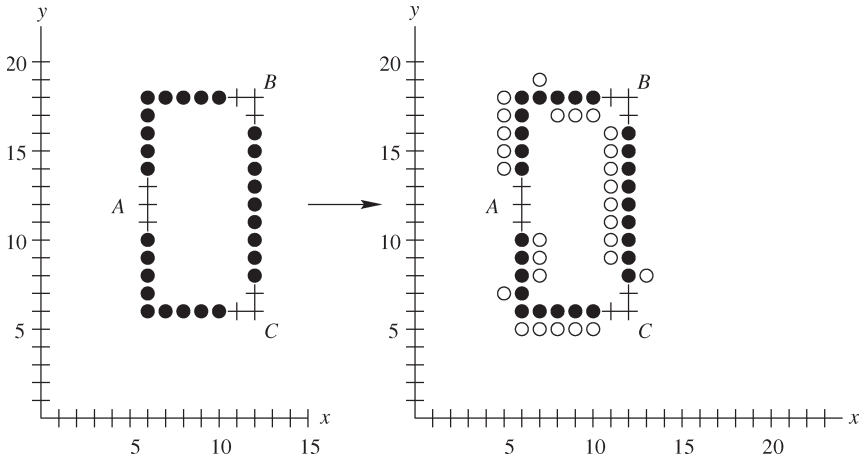


Fig. 14. Step 4 of the reduction. Black nodes are nodes in P', and P_l nodes are white. Nodes denoted with a "+" are in node regions.

four nodes in P along this POGD segment (starting from the center node) as n_1, n_2, n_3, and n_4. Hence, n_1 corresponds to a center node, and n_2 to an end node. Adjust the P_l nodes corresponding to n_3 and n_4, so that they are not within one grid length of each other. Note that this adjustment to the CDSUDG instance can be done for any P_l, while still satisfying the other conditions required of nodes in P_l. Therefore, the CDSUDG instance is still valid after this adjustment has been made, and all proofs concerning the CDSUDG instance [5] still hold for this modified reduction.

1) The nodes of the BCAST instance are the same as those in the generated CDSUDG instance (note that this is valid because each node in the generated CDSUDG instance is located at integer coordinates).
2) Set the source node of the BCAST instance s to be n_3 from above. This is demonstrated in Fig. 15, where the source is chosen to be at (12, 8).
3) The range of each BCAST node is set to one grid length.

Note that even with the addition of these steps, the total time for the reduction is still polynomial.

2) Proving NP-Hardness From This Reduction: Assume that we have taken a PLANAR instance and converted it to an instance of CDSUDG, and extended the CDSUDG instance

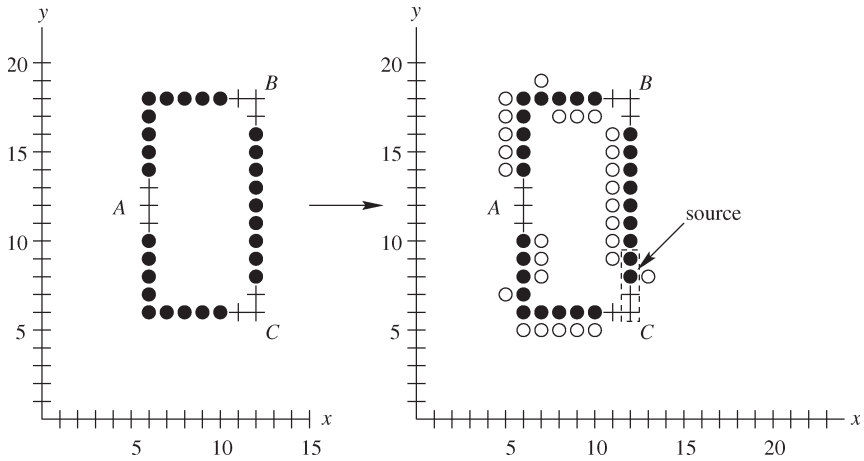


Fig. 15. Step 4 of the reduction, with a box around the nodes n_1 through n_4. Black nodes are nodes in P', and P_l nodes are white. Nodes denoted with a "+" are in node regions.

as noted above to construct an instance of BCAST. Then, the following lemma holds.

Lemma 2: There exists a BCAST tree of power no more than $|V| - |E| - 1 + k + |P''|$ iff there exists a connected node cover of size no more than k in the original instance of PLANAR.

Proof: Note the following observations about the BCAST instance constructed.

Observation 1: All neighbors of a node in the instance of CDSUDG are at distance exactly 1. Because the range of each node in the BCAST instance is one, this implies that in any BCAST tree, a given node is either using one unit of power or zero units of power. Therefore, we can consider a node in the BCAST instance as either being "ON" or "OFF."

Observation 2: The source node must be included in any connected dominating set in the generated instance of CDSUDG (because it is the sole node within one grid length of its corresponding P_l node).

Observation 3: Any connected dominating set (CDS) for the instance of CDSUDG can be mapped to a valid tree in the matching BCAST problem. To do so, turn on only those nodes in the BCAST instance that are in CDS. This is a valid BCAST tree, because it includes the source s as turned "ON" (by Observation 2), and for a given node n in the BCAST instance, there is a path from s to that node via "ON" nodes (by virtue of CDS being a connected dominating set). Additionally, the number of elements in CDS is equal to the power used in the BCAST instance (by Observation 1). Therefore, every solution to the generated CDSUDG instance maps to a corresponding BCAST solution. We can also prove the converse statement. To prove this, note that the "ON" nodes in a BCAST solution must constitute a dominating set (otherwise, there is a node which cannot be reached by the source in the BCAST solution, implying it is invalid). Additionally, in any valid BCAST tree, there is a path from the source to every "ON" node. This implies the set of "ON" nodes is also connected. Therefore, we can map a BCAST solution to a CDS in the matching CDSUDG problem by selecting the set of "ON" nodes. Note, that the power used in the BCAST solution is exactly equal to the cardinality of the CDS that it maps to.

Observation 3 implies that there exists a connected dominating set of size no more than J in the CDSUDG instance iff the corresponding instance of BCAST contains a broadcast tree of power no more than J. This statement, taken together with Lemma 1, implies Lemma 2. ■

Lemmas 1 and 2 imply that we can map every instance of PLANAR to an instance of BCAST in polynomial time, proving that BCAST is indeed NP-hard. Also, note that the coordinates of each BCAST node generated this way have sizes that are, at most, a polynomial function in the number of nodes (because in [4], the size of the POGD is polynomial in the number of nodes). This fact further implies that BCAST is strongly NP-hard. For problems in which a value is a parameter (in this case, the value is node coordinates), a strongly NP-complete problem is one that remains NP-complete even if the parameters are restricted to be polynomially bounded by the size of the problem instance [2].

In order to complete our proof, we must show that BCAST ϵ NP. This is clearly true because verifying the correctness of a broadcast tree can be done in polynomial time.

REFERENCES

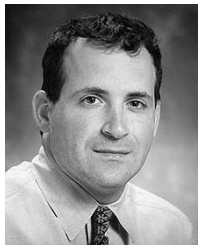
[1] E. Royer and C. K. Toh, "A review of current routing protocols for ad hoc wireless networks," *IEEE Pers. Commun.*, vol. 6, no. 2, pp. 46–55, Apr. 1999.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[3] ——, "The rectilinear Steiner problem is NP-complete," *SIAM J. Appl. Math.*, vol. 32, no. 4, pp. 826–834, Jun. 1977.
[4] R. Tamassia and I. G. Tollis, "Planar grid embedding in linear time," *IEEE Trans. Circuits Syst.*, vol. 36, no. 9, pp. 1230–1234, Sep. 1989.
[5] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," *Discrete Math.*, vol. 86, no. 1–3, pp. 165–177, 1990.
[6] W. T. Chen and N. F. Huang, "The strongly connecting problem on multihop packet radio networks," *IEEE Trans. Commun.*, vol. 37, no. 3, pp. 293–295, Oct. 1989.
[7] P. A. Humblet, "A distributed algorithm for minimum weight directed spanning trees," *IEEE Trans. Commun.*, vol. 31, no. 6, pp. 756–762, Jun. 1983.
[8] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," in *Proc. IEEE Conf. Computer Communications (INFOCOM)*, Tel Aviv, Israel, Mar. 2000, pp. 585–594.
[9] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *Proc. Int. Conf. Communications (ICC)*, Montreal, QC, Canada, 1997, pp. 376–380.
[10] B. Daneshrad, *Variable Rate Low Power FHSS Baseband Processor*. [Online]. Available: http://www.ee.ucla.edu/~babak/thss/thss.html
[11] M. B. Pursley, H. B. Russell, and J. S. Wysocarski, "Energy-efficient transmission and routing protocols for wireless multiple-hop networks and spread-spectrum radios," in *AFCEA/IEEE EuroComm Conf.*, Munich, Germany, May 17–19, 2000, pp. 1–5.
[12] N. Jindal and A. Goldsmith, "Capacity and optimal power allocation for fading broadcast channels with minimum rates," in *IEEE Global Telecommunications (GLOBECOM)*, San Antonio, TX, 2001, pp. 1292–1296.
[13] S. Kandukuri and S. Boyd, "Optimal power control in interference limited fading wireless channels with outage probability specifications," *IEEE J. Sel. Areas Commun., Wirel. Commun. Series*, vol. 1, no. 1, pp. 46–55, 2001.
[14] K. Leung, "Integrated link adaptation and power control for wireless IP networks," in *Proc. IEEE Vehicular Technology Conf. (VTC)*, Tokyo, Japan, May 2000, pp. 2086–2092.
[15] P.-J. Wan, G. Calinescu, X.-Y. Li, and O. Frieder, "Minimum-energy broadcast routing in static ad hoc wireless networks," in *Proc. IEEE Conf. Computer Communications (INFOCOM)*, Anchorage, AK, 2001, vol. 2, pp. 1162–1171.
[16] V. Rodoplu and T. H. Meng, "Minimum energy mobile wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 8, pp. 1333–1344, Aug. 1999.
[17] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1991.
[18] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan, "A cluster-based approach for routing in dynamic networks," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, 1997, vol. 27, pp. 49–64.

Ashwinder S. Ahluwalia received the B.S. and M.Eng. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, in 2002.

His body of research work deals primarily with graph-optimization problems in wireless ad hoc networks, concerning both complexity theoretic results and algorithms for approximation. He is currently a Senior Member of Technical Staff at Oracle Corporation in the Distributed Database Group. At Oracle, he primarily works on networking and systems problems that arise from the navigation of large data sets in multitiered database applications.



Eytan H. Modiano (S'90–M'93–SM'00) received the B.S. degree in electrical engineering and computer science from the University of Connecticut at Storrs in 1986, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, in 1989 and 1992, respectively.

He was a Naval Research Laboratory Fellow between 1987–1992 and a National Research Council Post-Doctoral Fellow between 1992–1993. Between 1993–1999, he was the Project Leader for the Massachusetts Institute of Technology (MIT) Lincoln Laboratory's Next Generation Internet (NGI) project. Since 1999, he has been an Associate Professor in the Department of Aeronautics and Astronautics and the Laboratory for Information and Decision Systems (LIDS) at MIT. His research is on communication networks and protocols with emphasis on satellite, wireless, and optical networks.

Dr. Modiano had served as a Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC) Special Issue on Wavelength Division Multiplexing (WDM) Network Architectures; the *Computer Networks Journal* Special Issue on Broadband Internet Access; the *Journal of Communications and Networks* Special Issue on Wireless Ad-Hoc Networks; and for IEEE JOURNAL OF LIGHTWAVE TECHNOLOGY Special Issue on Optical Networks. He is currently an Associate Editor for Communication Networks for the IEEE TRANSACTIONS ON INFORMATION THEORY and for *The International Journal of Satellite Communications*. He is the Technical Program Co-Chair for Wiopt 2006 and Infocom 2007.