Floréen, Patrik; Kaski, Petteri; Kohonen, Jukka; Orponen, Pekka

Multicast time maximization in energy constrained wireless networks

Published in:
DIALM-POMC '03 Joint Workshop on Foundations of Mobile Computing, San Diego, CA, USA, 19.9.2003

DOI:
10.1145/941079.941087

Published: 01/01/2003

Document Version
Peer reviewed version

Multicast Time Maximization in Energy Constrained Wireless Networks*

Patrik Floréen
Helsinki Institute for Information Technology
Basic Research Unit
P.O. Box 26
FIN-00014 University of Helsinki, Finland
patrik.floreen@cs.helsinki.fi

Petteri Kaski
Laboratory for Theoretical Computer Science
P.O. Box 5400
FIN-02015 Helsinki University of Technology, Finland
petteri.kaski@hut.fi

Jukka Kohonen
Helsinki Institute for Information Technology
Basic Research Unit
P.O. Box 26
FIN-00014 University of Helsinki, Finland
jukka.kohonen@cs.helsinki.fi

Pekka Orponen
Laboratory for Theoretical Computer Science
P.O. Box 5400
FIN-02015 Helsinki University of Technology, Finland
pekka.orponen@hut.fi

ABSTRACT

We consider the problem of maximizing the lifetime of a given multicast connection in a wireless network of energy-constrained (e.g. battery-operated) nodes, by choosing ideal transmission power levels for the nodes relaying the connection. We distinguish between two basic operating modes: In a *static* assignment, the power levels of the nodes are set at the beginning and remain unchanged until the nodes are depleted of energy. In a *dynamic* assignment, the powers can be adjusted during operation.

We show that lifetime-maximizing static power assignments can be found in polynomial time, whereas for dynamic assignments, a quantized-time version of the problem is **NP**-hard. We then study the approximability of the quantized dynamic case and conclude that no polynomial time approximation scheme (PTAS) exists for the problem unless $\mathbf{P} = \mathbf{NP}$. Finally, by considering two approximation heuristics for the dynamic case, we show experimentally that the lifetime of a dynamically maintained multicast connection can be made several times longer than what can be achieved by the best possible static assignment.

*Research supported by the Academy of Finland, Grants 202203 (J. Kohonen and P. Floréen) and 202205 (P. Kaski and P. Orponen); and by the Foundation of Technology, Helsinki, Finland (Tekniikan Edistämissäätiö) (P. Kaski).

DIALM–POMC'03, September 19, 2003, San Diego, California, USA.

Categories and Subject Descriptors

G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless communication*

General Terms

Algorithms, Performance, Theory

Keywords

optimization, energy saving, multi-hop packet networks, wireless networks, sensor networks, multicasting, Steiner tree packing

1. MOTIVATION AND STATEMENT OF THE PROBLEM

Wireless "ad hoc" communication networks, consisting of a collection of radio transceivers with no prearranged infrastructure, have been studied intensively during the past few years [21]. The general area of *topology control* in ad hoc networks [23] is concerned with assigning appropriate transmission power levels to the nodes so that some desired topological property holds in the induced transmission graph. A central problem in this area is the maintenance of a *multicast connection* from a given source node to a group of sink nodes.

Battery power is a serious limiting constraint in many applications of ad hoc networks, and accordingly much attention has been paid to energy-efficient designs in this area [12]. In a wireless node, power is used for transmitting and receiving data, internal data processing, and simply for being "on" in an idle mode. The power required for transmission and reception, however, dominates [15, 27]. For simplicity, we consider only the power required for transmission. A similar model is considered in e.g. the papers [15, 17, 18].

We model an ad hoc network as a directed graph with the transceivers as nodes. Node i can communicate directly

to node j, if the transmission power from i exceeds some threshold value d_{ij} [17, 24]. As a consequence, when the power is high enough to reach a certain node, some other nodes may be reached simultaneously with the same transmission. In other words, the minimum "cost" of transmission from node i to nodes $j_1, j_2, \ldots, j_m$ is $\max_k d_{ij_k}$ rather than $\sum_k d_{ij_k}$. With a particular setting of transmission powers for the nodes, node j may transmit with power different from i, so a direct connection from i to j does not imply that direct communication from j to i is possible. Note that this is different from the symmetric model employed in e.g. [1].

In reality, the actual transmission powers and threshold values d_{ij} depend on several environmental and technological factors [24], as well as situations of use (e.g. QoS requirements), but these go for the most part into the computation of proper d_{ij} values in the given setting, and therefore our model is independent of such technical details. A common approximation is to choose $d_{ij} \propto {r_{ij}}^{\alpha}$, where r_{ij} is the physical distance between nodes i and j, and the exponent α, $2 \leq \alpha \lesssim 4$, models the decay of the radio signal in the intervening medium. (A thorough description of mobile radio propagation and transmission/reception technology can be found in [25].)

In multicasting, one of the nodes, usually indexed as 1, is a designated *source* node, and some k other nodes are the designated *sinks*. The requirement is to keep the source connected to all the sinks.

A number of recent papers (e.g. [5, 6, 8, 9, 16, 18, 19, 20, 28, 29, 30]) have considered the topic of energy-aware *broadcasting* in wireless networks, i.e. the problem of maintaining a transmission path from a given source node to all the other nodes. However, with rare exceptions such as [18, 20], all of these works (as well as many older ones, e.g. [14, 26]), take it as their goal to minimize the *total power consumption* of the entire network, i.e. they address the wireless analogue of the classical problem of finding a minimal spanning tree for a network. (Interestingly, in wireless networks this "minimum power broadcast tree" problem turns out to be **NP**-complete, as proved in e.g. [5, 8, 16].) Some of the papers, e.g. [16, 19, 29, 30] address also the problem of constructing minimum multicast trees under the same metric.

However, minimal total power consumption does not guarantee maximum lifetime for a network (either for broadcast or multicast), as has been noted for instance in [7]. Our objective is also different from minimizing the maximum power applied at any individual node, as considered in [17, 24], because we take into account that different nodes have different energy supplies from the outset. A number of different power-aware metrics, including the ones above, is given in [27].

Previous work closest to the present paper consists of articles [17] and [18]. Our network model and some fundamental lemmas are adapted from [17], where the authors consider e.g. the task of minimizing the maximum power required at any individual node in order to achieve some desired topological property. In [18], the problem of maximizing broadcast lifetimes is considered in both the static and dynamic power assignment settings; however the paper contains neither theoretical analysis of the complexity of the problem, nor experimental results validating the chosen heuristics.

We are now ready for some definitions.

DEFINITION 1. *A* **power threshold graph** $G = (V, A, d)$ *is a complete directed graph with nodes* $V = \{1, \ldots, n\}$ *and threshold power values* $d_{ij} \in \mathbb{R} \cup \{\infty\}$ *on each arc* $(i, j) \in A = V \times V$.

Note that we include ∞ among the possible values to indicate that direct communication is impossible (due to, e.g., an impenetrable medium between the nodes). We can trivially take d_{ii} to be 0 for all i.

Each node i carries a battery with limited energy supply e_i. Disregarding the details of battery technology, transmission at power p for a time t consumes energy $w = pt$. More precisely, the energy consumed by node i from the starting point up to time t is $w_i(t) = \int_0^t p_i(\tau)\, d\tau$, where $p_i(\tau) \geq 0$ is the transmission power of node i at time τ. The transmission powers we consider are piecewise constant with a finite number of discontinuities.

DEFINITION 2. *A* **power assignment** $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ *in a power threshold graph* G *of* n *nodes associates to each node in* G *its transmission power value. Since the power applied at a node may in general change over time, each* p_i *(and hence* $\mathbf{p}$*) is actually a function of time* t*. A power assignment that stays constant over time is* **static**. *(More precisely, a static assignment provides for each node* i *a power value* c_i *and time* t_i *so that* $p_i(t) = c_i$ *for* $0 \leq t < t_i$ *and* $p_i(t) = 0$ *for* $t \geq t_i$.*) More general power assignments are* **dynamic**.

DEFINITION 3. *The* **transmission graph** $G_t^{\mathbf{p}}$ *induced by a particular power assignment* $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ *at time* $t \geq 0$ *is the subgraph obtained from the power threshold graph* G *by including only those arcs* (i, j) *for which* $p_i(t) \geq d_{ij}$.

Note that we use the term "transmission graph" as in [23], and differently from [1] where this term is used for a symmetric power threshold graph.

DEFINITION 4. *The* **lifetime** *of a node* $i \in V$ *with initial energy supply* e_i *is* $t_i = \sup\{\, t \mid w_i(t) < e_i \,\}$*, where* $w_i(t)$ *is the energy consumed at node* i *up to time* t*. During the time interval* $[0, t_i[$*, the node is* **alive**. *A node that is not alive is* **dead**.

DEFINITION 5. *The* **multicast connection time** *for a set of* **sink nodes** $U \subseteq V$*, with respect to* **source node** 1*, in a power threshold graph* G *subject to a power assignment* $\mathbf{p}$ *is the supremum over* $t \geq 0$ *such that for each* $i \in U$ *there is a directed path connecting* 1 *to* i *in all graphs* $G_\tau^{\mathbf{p}}$ *for* $\tau \leq t$.

In other words, the multicast connection time for a set of sink nodes U is the maximum continuous amount of time starting from instant $t = 0$ such that there are connections in $G_t^{\mathbf{p}}$ from source 1 to all the nodes in U. As a limiting case, e.g. if U is empty, the multicast connection time can also be infinite.

DEFINITION 6. *A power assignment* $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ *is* **feasible** *with respect to a given* **energy supply** $\mathbf{e} = (e_1, e_2, \ldots, e_n)$*, if the energy expenditures determined by* $\mathbf{p}$ *satisfy* $w_i(t) \leq e_i$ *for all* $i \in V$ *and* $t \geq 0$.

DEFINITION 7. *The* **multicast time maximization problem** *for a given power threshold graph* $G = (V, A, d)$*, energy supply* $\mathbf{e} = (e_1, \ldots, e_n)$*, and set of sink nodes* $U \subseteq V$ *is to determine a feasible power assignment* $\mathbf{p}$ *that maximizes the multicast connection time for* U.

Note that several of the nodes may run out of energy before the source does so. Ensuring the connectivity from the source to the sinks with the help of other nodes is analogous to finding so called *Steiner trees* [11, 22] in undirected graphs.

DEFINITION 8. *A* **Steiner tree** *in an undirected graph $G = (V, E)$ with a set of* **critical nodes** *$W \subseteq V$ is a subtree T of G that includes all the nodes in W.*

The rest of the paper is organized as follows. In Section 2, we study multicast time maximization by static power assignments and provide a polynomial time algorithm for this problem, drawing on the results of [17]. In Section 3, we turn to general dynamic power assignments and show that a quantized-time version of the multicast time maximization now becomes **NP**-hard. The lemma showing that (our version of) the underlying "Steiner tree packing problem" is **NP**-complete is of interest in itself. We also study the approximability of the dynamic case and conclude that no polynomial time approximation scheme (PTAS) exists for the quantized-time version of the problem unless $\mathbf{P} = \mathbf{NP}$. In Section 4 we suggest two randomized approximation methods for finding good dynamic assignments, and in Section 5 we report on simulation experiments. We conclude with a summary and some ideas for further research in Section 6.

2. STATIC POWER ASSIGNMENTS

The following definition is from [4, 17].

DEFINITION 9. *A graph property $\mathbb{P}$ is* **monotone***, if adding edges to a graph can never make $\mathbb{P}$ change from true to untrue. In terms of transmission graphs, this means that increasing power at nodes preserves the property.*

The property we consider in this paper is the existence of directed paths from source node 1 to all sink nodes $i \in U$. This **multicast property** is clearly monotone.

The following definition is a generalization of the **MaxP** problem given in [17].

DEFINITION 10. *Given a power threshold graph $G = (V, A, d)$, let f_i be for each node $i \in V$ some quantity that is monotonically increasing in the power applied at node i. The problem* **Max$f$** *with respect to property $\mathbb{P}$ is to determine static f_i-values for all the nodes $i \in V$ in such a way that the corresponding transition graph has property $\mathbb{P}$, and the maximum of the f_i-values at the nodes is minimized.*

The following is a slight generalization of the fundamental lemma (Lemma 4.1) in [17], in which the f_i-values considered are exactly the transmission powers at the nodes.

LEMMA 1. *For any instance of the* **Max$f$** *problem with respect to a monotone $\mathbb{P}$, there is an optimal solution in which the f_i-values at each node are equal.*

PROOF. As the f_i-values are monotonically increasing in the power values, the monotonicity of property $\mathbb{P}$ can equally well be considered with respect to the f_i-values as with respect to the power values.

Consider an optimal solution to the given instance where the nodes do not necessarily have the same f_i-values. Let Q denote the maximum f_i-value assigned to any node i. Since the property $\mathbb{P}$ is monotone, for any node whose f_i-value is less than Q, we can increase it to Q without destroying the property. □

Now we can prove the following straightforward generalization of Theorem 4.1 in [17]. For completeness, we include the proof, even though it includes only small modifications to the proof given in [17].

THEOREM 1. *For any monotone graph property $\mathbb{P}$ that can be tested in polynomial time, the* **Max$f$** *problem can be solved in polynomial time.*

PROOF. For any instance of the problem, there exists by the preceding lemma an optimal solution with equal f_i-values at the nodes. The basic idea is that there are only a polynomial number of possible f_i-values and we can actually search through this set of possible values efficiently using, e.g. binary search.

Consider any node $i \in V$. The number of different power values that need to be considered for i is at most n, since at most one new power value is needed for each node $j \neq i$ to ensure the communication from i to j, namely d_{ij}. Therefore, for all of the n nodes, the total number of corresponding candidate f_i-values to be considered is at most n^2.

Each candidate f_i-value corresponds to a power assignment, and the induced directed transmission graph can be constructed in $O(n^2)$ time. Let $F_{\mathbb{P}}(n)$ denote the time needed to test whether property $\mathbb{P}$ holds for a directed graph with n nodes. Thus, the time needed to test whether property $\mathbb{P}$ holds for each candidate solution value is $O(n^2 + F_{\mathbb{P}}(n))$. An optimal solution can be obtained by sorting the $O(n^2)$ candidate solution values and using binary search to determine the smallest f_i-value for which property $\mathbb{P}$ holds. Since the number of candidate solution values is $O(n^2)$, the time taken by the sorting step is $O(n^2 \log n)$. The binary search would try $O(\log n)$ candidate solution values and the time spent for testing each candidate is $O(n^2 + F_{\mathbb{P}}(n))$. Thus, the total running time of this algorithm is $O((n^2 + F_{\mathbb{P}}(n)) \log n)$, i.e. polynomial. □

From this proof we learn that we need consider only a discrete set of possible power values, i.e. those that coincide with some d_{ij}. The same idea will appear in the algorithms for the dynamic case in Section 4.

COROLLARY 1. *The multicast time maximization problem constrained to static power assignments can be solved in polynomial time.*

PROOF. For a constant power p_i, the lifetime of node i is $t_i = e_i/p_i$, a decreasing function of p_i. Take as the f_i-value at node i the negative of its lifetime. This is clearly a monotonically increasing function of the power p_i. Moreover, maximizing the lifetime t_i is equal to minimizing f_i. The situation of Lemma 1 corresponds to all nodes being alive for exactly the same time and then running out of energy simultaneously. □

3. DYNAMIC POWER ASSIGNMENTS

We now turn our attention to the more difficult task of multicast time maximization using dynamic power assignments. For simplicity, we assume that the possible control times (discontinuities) for the power assignment are quantized to occur at intervals of one time unit.[1]

[1]While this is a natural assumption, it remains a possibility that one can achieve more efficient power assignments by applying arbitrarily fine-grained controls.

We first give a direct combinatorial proof showing that the multicast time maximization problem under quantized dynamic power assignments is **NP**-hard. For this we need the following lemma, which is of interest in itself. Note that the term "Steiner tree packing" as used in the literature, e.g. in [3], refers to a more general problem that could more appropriately be called Steiner forest packing: the set of critical nodes to be included in the Steiner tree are predetermined, but may be different for each tree.

DEFINITION 11. [11, p. 221] *The* **3-dimensional matching problem (3DM)** *is the following. Given a set $T \subseteq X \times Y \times Z$, where $|X| = |Y| = |Z| = q$, find a* **matching** *M for T, i.e. a subset $M \subseteq T$ such that $|M| = q$ and no two distinct elements in M agree in any coordinate.*

DEFINITION 12. *The* **node-disjoint (edge-disjoint) Steiner tree packing problem** *is the following. Given an undirected graph $G = (V, E)$, a set of critical nodes $W \subseteq V$, and a positive integer N, decide whether there exist at least N pairwise node-disjoint (edge-disjoint) Steiner trees in G. Note that only the noncritical nodes are required to be node-disjoint; the critical nodes are to be included in all the Steiner trees.*

LEMMA 2. *The node-disjoint Steiner tree packing problem is* **NP***-complete.*

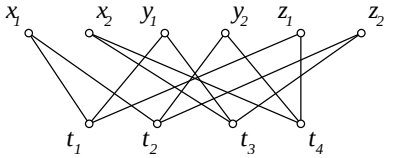
PROOF. Suppose an instance of the node-disjoint Steiner tree packing problem is given, where $G = (V, E)$ is the input graph, $W \subseteq V$ is the set of critical nodes, and N is the required number of node-disjoint Steiner trees. A trivial upper bound for the number of node-disjoint subgraphs in G is $|V| + 1$. Hence, the problem is in **NP** because a nondeterministic algorithm can guess N subgraphs of G and check in deterministic polynomial time that these subgraphs form a collection of node-disjoint Steiner trees for W.

We show completeness by transformation from 3DM. Let $T \subseteq X \times Y \times Z$ be an instance of 3DM with $|X| = |Y| = |Z| = q$ and $|T| = m$. Without loss of generality we may assume that $m \geq q$ and that the sets X, Y, Z, T are pairwise disjoint. Denote the elements of X by $x_1, \ldots, x_q$ and similarly for the other sets Y, Z, T.

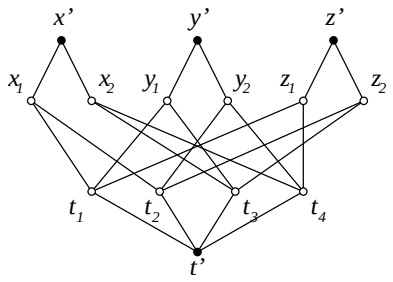
To make the proof more readable, we shall consider an example alongside the formal treatment. In the example, let $X = \{x_1, x_2\}, Y = \{y_1, y_2\}, Z = \{z_1, z_2\}$ and $T = \{t_1, t_2, t_3, t_4\}$, where

$$t_1 = (x_1, y_1, z_1), \quad t_2 = (x_1, y_2, z_2),$$
$$t_3 = (x_2, y_1, z_2), \quad t_4 = (x_2, y_2, z_1).$$

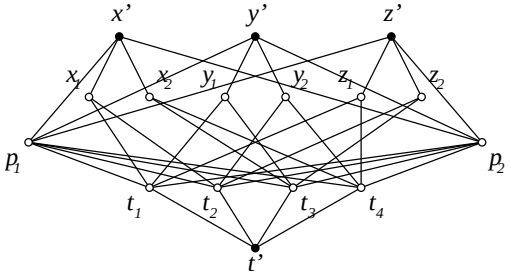
We construct the input graph $G = (V, E)$ and the set of critical nodes for the Steiner tree packing instance from the 3DM instance in three steps. In the first step, let the node set of the input graph be $V = X \cup Y \cup Z \cup T$. All of these nodes are noncritical. Now, for every $\ell = 1, \ldots, m$, we add the three edges $\{x_i, t_\ell\}, \{y_j, t_\ell\}, \{z_k, t_\ell\}$ to the graph, where x_i, y_j, z_k are determined from $t_\ell = (x_i, y_j, z_k)$. The graph resulting from the example instance after the first step is depicted below.



In the second step, we insert four critical nodes x', y', z', t' into the graph, and join these by an edge to all of the nodes in the respective sets X, Y, Z, T; that is, x' is adjacent to all the nodes in X, y' is adjacent to all the nodes in Y, and so on.



In the third step, we insert $m - q$ noncritical nodes $P = \{p_1, \ldots, p_{m-q}\}$ into the graph, and join each of these by an edge to the critical nodes x', y', z' and to all of the nodes in T.



This completes the description of G. The transformation is clearly computable in polynomial time. We now show that G contains m pairwise node-disjoint Steiner trees if and only if there exists a matching M for T.

We start with the "if" direction. Suppose $M = \{t_{\ell_1}, \ldots, t_{\ell_q}\} \subseteq T$ is a matching for T. We can construct q pairwise node-disjoint Steiner trees in G by taking for each $s = 1, \ldots, q$ the subgraph induced by the nodes $x', y', z', t', t_{\ell_s}, x_{i_s}, y_{j_s}, z_{k_s}$, where $x_{i_s}, y_{j_s}, z_{k_s}$ are determined from $t_{\ell_s} = (x_{i_s}, y_{j_s}, z_{k_s})$. The remaining $m - q$ trees can be constructed by taking the subgraph induced by x', y', z', t', and any two nodes p_i and t_ℓ not used so far.

It remains to prove the "only if" direction. Suppose we are given m node-disjoint Steiner trees in G. Because each of the critical nodes x', y', z', t' is adjacent to exactly m nodes, which are noncritical nodes, we must have that each of the Steiner trees contains exactly one node from T and either (i) exactly one node from P or (ii) exactly one node from each of the sets X, Y, Z. We can construct a matching for T from the nodes $t_\ell \in T$ in the q trees in which case (ii) occurs. □

THEOREM 2. *The multicast time maximization problem under quantized dynamic power assignments is* **NP***-hard.*

PROOF. We transform an instance G, W, N of the node-disjoint Steiner tree packing problem into an instance of the multicast time maximization problem. Without loss of generality we may assume that the critical nodes W are pairwise nonadjacent in G. First, replace every undirected edge $\{i, j\}$ in G with two directed edges (i, j), (j, i) to obtain a directed graph. Then, put $d_{ij} = 1$ if (i, j) is an edge; otherwise $d_{ij} = \infty$. Select (arbitrarily) one of the critical

nodes W as the source node; the other critical nodes in W become the sink nodes. To complete the transformation, put $e_i = N$ for the source/sink nodes and $e_i = 1$ for all the other nodes. It is easy to see that the transformed instance has a feasible quantized-time power assignment achieving multicast time N if and only if the original instance has N node-disjoint Steiner trees. □

Clearly, the above proof of Theorem 2 does not apply to the general case where the discontinuities in the power assignments are not quantized to occur at intervals of one time unit.

We now turn to study the approximability of the multicast time maximization problem. We show that the multicast time maximization problem is hard for the complexity class **APX**, which consists of all **NP** optimization problems that admit a polynomial time approximation algorithm that achieves a constant performance ratio (see [2]). In particular, this implies that the multicast time maximization problem under quantized dynamic power assignments does not admit a polynomial time approximation scheme (PTAS) unless $\mathbf{P} = \mathbf{NP}$.

We base our **APX**-hardness proof on the following result.[2]

THEOREM 3. [13, Corollary 4.3] *The edge-disjoint Steiner tree packing problem is* **APX***-hard.*

LEMMA 3. *There exists an approximation preserving polynomial time reduction from the edge-disjoint Steiner tree packing problem to the node-disjoint Steiner tree packing problem.*

PROOF. Let $G = (V, E)$ be an instance of the edge-disjoint Steiner tree packing problem, where $W \subseteq V$ is the set of critical nodes. We transform this instance into an instance of the node-disjoint Steiner tree packing problem in two steps.

In the first step, we perform the following local transformation for each noncritical node $v \in V \backslash W$. Let v be incident with k edges $e_1, \ldots, e_k$. We remove v from the graph and replace it with k new noncritical nodes $v_1, \ldots, v_k$ so that, for every $i = 1, \ldots, k$, the edge e_i becomes incident with the node v_i. We then connect the nodes v_i pairwise so that they form a k-clique. For purposes of analysing the reduction later, we color all the edges e_i blue and the edges in the k-clique green.

In the second step, we subdivide each edge joining two critical nodes into two edges by inserting a new noncritical node in the middle; these two edges are colored red. This completes the description of the transformation, which is clearly computable in polynomial time. We claim that the input instance contains N edge-disjoint Steiner trees if and only if the transformed instance contains N node-disjoint Steiner trees.

We can transform a Steiner tree T in the edge-disjoint instance into a Steiner tree T' in the node-disjoint instance as follows. First, for each edge in T, we include the corresponding blue edge (or both of the corresponding red edges) into T'. We then connect the blue edges into each other by a minimal number of green edges. This transformation maps edge-disjoint trees to node-disjoint trees because distinct blue edges cannot have a common noncritical endpoint.

Conversely, we can transform a Steiner tree T' in the node-disjoint instance into a Steiner tree T in the edge-disjoint instance by taking all the blue and red edges in T' and inserting the corresponding edges into T. There is one exception: if only one red edge from a pair of red edges occurs in T', then we do not insert the corresponding edge into T because this would create a cycle. This transformation maps node-disjoint trees map to edge-disjoint trees because each blue edge occurs in at most one tree in a collection of node-disjoint trees. Furthermore, the red edges from a pair of red edges cannot occur in different trees of a node-disjoint collection because the edges share a noncritical endpoint. □

Combining Theorem 3 and the reduction in Lemma 3, we obtain:

COROLLARY 2. *The node-disjoint Steiner tree packing problem is* **APX***-hard.*

COROLLARY 3. *The multicast time maximization problem under quantized dynamic power assignments is* **APX***-hard.*

PROOF. The transformation from node-disjoint Steiner tree packing into multicast time maximization in the proof of Theorem 2 is approximation preserving. □

It remains an open problem whether the node-disjoint Steiner tree packing problem or the multicast time maximization problem belong to the class **APX**, i.e. possess polynomial time approximation algorithms with a constant performance ratio.

4. ALGORITHMS FOR THE DYNAMIC POWER ASSIGNMENT PROBLEM

We present two randomized algorithms for the multicast time maximization problem under dynamic power assignments, and a method for bounding the multicast connection time from above.

The first algorithm is based on the static algorithm presented in Section 2. In the static solution, all nodes will run out of energy simultaneously. Our algorithm **RND-GREEDY** (see box) reduces node powers in a random order to save energy for further multicast trees. As a randomized algorithm, it produces varying results, so it is useful to iterate it several times and choose the best solution found.

Although this relatively simple algorithm routinely gives dynamic solutions with multicast times that are 2 to 4 times the static one, even better performance can be obtained. For this, the dynamic power assignment problem is split into two essentially separate subproblems: *sampling* a good collection from the set of all viable static power assignments, and *scheduling*, i.e. deciding for how long each assignment in the collection should be used. By a **viable** static power assignment we mean one that fulfills the multicast property. We shall tackle the scheduling problem first.

A collection of m viable static power assignments can be concisely represented as columns of an $n \times m$ matrix $\mathbf{P}$, where $\mathbf{P}_{ij}$ indicates the power of node i in the jth assignment. Now scheduling is a linear programming (LP) problem: Given $\mathbf{P}$ and an energy supply vector $\mathbf{e} = (e_1, \ldots, e_n)$, find a **schedule** vector $\mathbf{x} = (x_1, \ldots, x_m)$ that maximizes $\sum x_j$ (total multicast time), subject to $\mathbf{Px} \leq \mathbf{e}$ (energy constraints at all nodes), and $x_j \geq 0$ for all j. The dynamic power assignment is then as follows: for each $j = 1, \ldots, m$, run the jth power assignment for x_j time units. Note that

[2]We remark that only a proof sketch for the result is given in [13].

Algorithm RNDGREEDY

1. Choose powers $\mathbf{p} = (p_1, \ldots, p_n)$ with the polynomial time algorithm given by Corollary 1.
2. Repeat until the source has zero energy:
   2.1. For all nodes i in random order: Turn the power p_i as low as possible without breaking the multicast connectivity from the source to all sinks. (Perform a binary search over the values d_{ij} that lie between 0 and the p_i given by Step 1; for each value, check connectivity with depth-first search.)
   2.2. Run the network with this power assignment until some node runs out of energy. Update the energy supplies of all nodes according to the consumption.

Algorithm LPSCHEDULE

1. *Initialize:* $\mathbf{P} \leftarrow [\,]$, $\mathbf{e}_{red} \leftarrow \mathbf{e}$ (the true energy supply of each node).
2. Repeat for a number of iterations:
   2.1. *Sampling:* Generate a set of power assignments with **RNDGREEDY**, using reduced energy supply $\mathbf{e}_{red}$. Append all of them as new columns to $\mathbf{P}$.
   2.2. *Scheduling:* $\mathbf{x} \leftarrow$ optimum schedule for $\mathbf{P}$ with full energy supply $\mathbf{e}$ (solve as LP).
   2.3. *Consumption:* $\mathbf{w} \leftarrow \mathbf{P}\mathbf{x}$.
   2.4. *Energy for next iteration:* $r \leftarrow$ uniform random number in $[0, 1]$, $\mathbf{e}_{red} \leftarrow \mathbf{e} - r \cdot \mathbf{w}$.
3. Return the dynamic assignment given by $\mathbf{P}$ and $\mathbf{x}$.

here we treat time as a continuous quantity and not quantized to unit-length intervals as in Section 3.

This is a natural formulation of the dynamic power assignment problem in the following sense: If $\mathbf{P}$ contains all the viable static power assignments for a given network, scheduling will give the optimal dynamic assignment. However, for networks of nontrivial size, this is out of question, as the number of viable assignments is exponential in n. But for relatively large collections (e.g. $m \approx 1000$), the LP scheduling problem can be efficiently solved.

For *sampling*, simply generating a collection of individually good static power assignments is not enough. As pointed out in [18], such a collection is not necessarily very good for the dynamic power assignment. Instead, we would like to find static power assignments that exploit the energy supplies of different subsets of all nodes. This is the underlying idea in our second algorithm.

The algorithm **LPSCHEDULE** (see box) generates a collection $\mathbf{P}$ iteratively. New power assignments are obtained using **RNDGREEDY** and accumulated. Power assignments are never discarded from the collection; we rely on the scheduler to allocate zero time for inferior assignments. Scheduling is computationally cheap, as long as $\mathbf{P}$ remains reasonable in size. In our experiments, most of the running time of **LPSCHEDULE** is spent in generating new viable assignments using **RNDGREEDY**.

The trick is that the new assignments are encouraged to exploit other nodes than those heavily used by the current $\mathbf{P}$, by telling **RNDGREEDY** that only a part of the true energy supply is available. The strength of this encouragement varies according to a random factor $r \in [0, 1]$. Note that for $r = 0$, Step 2.1 of **LPSCHEDULE** reduces into another independent iteration of **RNDGREEDY** for the original problem. On the other hand, for $r = 1$, the new assignments are restricted to "leftover" nodes. Intermediate values of r provide a "soft" penalty for competing over energy with the current solution.[3]

Finally, we attack the problem from the opposite direction and formulate upper bounds for the dynamic multicast time in a given network. Such an upper bound can be used e.g. for assessing the quality of the solutions found by the algorithms.

Let the source node be 1, and its nearest neighbor 2. A trivial upper bound follows from the fact that the source has to reach at least its nearest neighbour, so $p_1 \geq d_{12}$. No matter how we choose the powers of other nodes, the multicast time cannot exceed $\frac{e_1}{d_{12}}$.

This idea can be extended to an arbitrary *cut* $C \subset V$ that contains the source, but not all the sinks. At all times, the power levels inside C must be large enough so that the transmission graph contains at least one path from the source to some node outside C. For small C, we can enumerate all such paths and find a time-maximizing schedule between them by solving an LP similar to that described above.

Experiments suggest that quite small cuts can give tight bounds on the dynamic multicast time. Indeed, for multicast trees in a large network, the bottleneck is in escaping a small neighbourhood of the source, constrained by its limited energy supply. After that, the large number of nodes in the rest of the network provides an abundance of alternative routes to the sinks.

5. EXPERIMENTS

The algorithms were implemented in MATLAB 6.5, using the Optimization Toolbox 2.2 for LP solving. Experiments were run on a workstation with a 1333 MHz AMD Athlon processor.

In the experiments, we place 100 nodes (1 source, 4 sinks) at random, uniformly distributed in the unit square. Power thresholds are computed as $d_{ij} = {r_{ij}}^2$. All nodes are given 1 unit of energy. An upper bound for the multicast time is computed using a cut consisting of the source and its six nearest neighbours.

An example network is shown in Fig. 1. The maximum static multicast time for this network is 59.7 units. Both dynamic algorithms were run for 100 iterations. Performance of the algorithms is compared in Fig. 2. The multicast time given by **RNDGREEDY** quickly reached about 3.3 times

[3] In fact, the randomization of r is not crucial, and a fixed value of e.g. 0.25 or 0.5 works quite well.

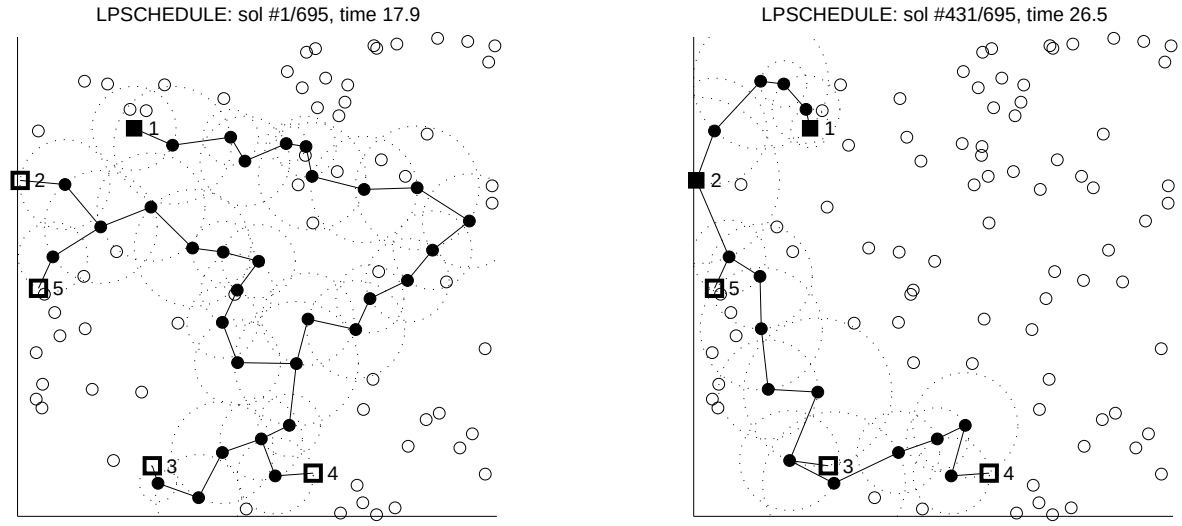


Figure 1: A randomly generated network, and multicast trees in two of the 695 power assignments generated by LPSCHEDULE for this network. Although the assignment on the left is also an optimal static solution and *could* be used for 59.7 units of time, the dynamic solution uses it for only 17.9 units. The assignment on the right is given the most time, 26.5 units. The full solution allocates nonzero time to 92 of the assignments, giving a dynamic multicast time of 236.8 units. Source (1) and sink nodes (2, 3, 4 and 5) are shown as squares. Filled nodes are transmitting at ranges indicated by the dotted circles, and hollow ones are idle.

the static solution, but remained constant after the first 39 iterations. Algorithm **LPSCHEDULE** generated 695 different power assignments, two of which are shown in detail in Fig. 1. The solution gives an improvement ratio of 3.96 over the static multicast time and reaches 92.6 % of the upper bound.

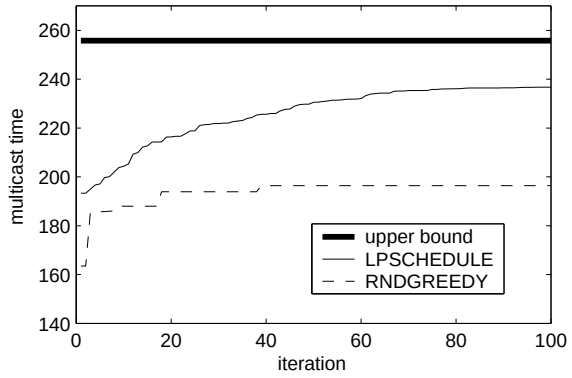


Figure 2: Progress of RNDGREEDY and LPSCHEDULE for the Fig. 1 network. (Total computation time 126 and 199 CPU seconds, respectively.)

To gain some idea of the performance of the algorithms in general, we next generated 50 random networks. For each network, we ran the static algorithm and the two dynamic ones, for 50 iterations each. The results are shown in Fig. 3. It can be seen that the dynamic power assignments gave multicast times which in general were about three times the static ones. Also, **LPSCHEDULE** often finds near-optimal solutions; in fact, in more than half of the networks (28 out of 50) the solution is within 1 % of the upper bound. Note that in one case the network was extreme in the sense that even the static solution reaches the upper bound, i.e., gives an optimal solution.

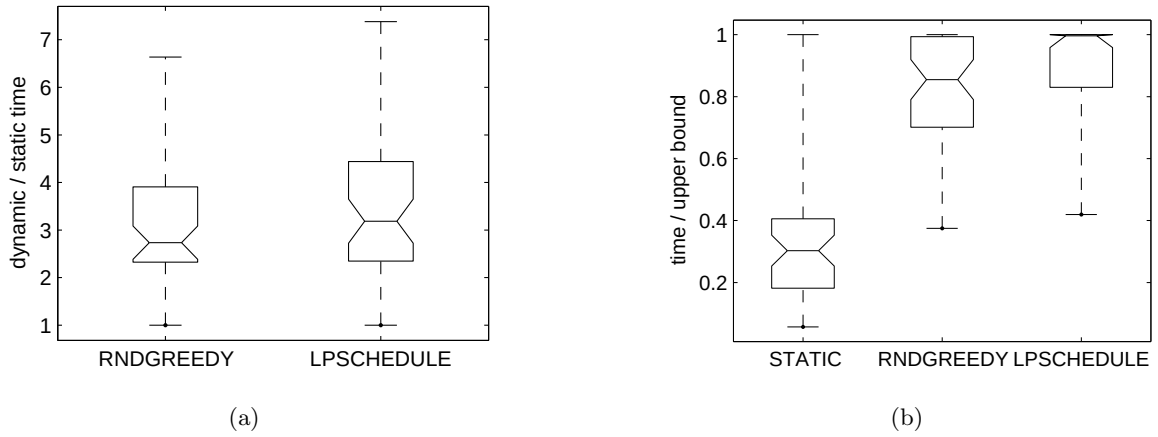
The results show that dynamic power assignments can be clearly superior to static ones and that our algorithm **LPSCHEDULE** can achieve very good solutions.

6. CONCLUSION

A large amount of current work has been directed towards energy minimization in wireless ad hoc networks, with the underlying goal of maximizing the lifetime. Our approach has been to optimize the multicast time directly. We have provided an optimal polynomial time algorithm for determining the maximum multicast time under the constraint that the transmission powers of all nodes are set to fixed values at the outset, and two approximation algorithms for finding good power assignment schedules when the powers at the nodes can be dynamically adjusted during operation. In fact, for small networks, the optimal dynamic multicast time can be determined using linear programming as long as the total number of all viable static power assignments is small enough, i.e. less than a few thousand.

We have also proved that finding optimal power assignment schedules in the dynamic case is **NP**-hard and thus not likely to be exactly solvable by a polynomial-time algorithm, when the time is quantized instead of real-valued. Furthermore, this problem is not likely to have a polynomial time approximation scheme. Whether the problem admits a constant performance ratio approximation algorithm remains an open problem. Also the computational complexity under real-valued time remains open.

We have assumed that the nodes are immobile, as in e.g.



(a) (b)

Figure 3: Distribution of multicast times, (a) relative to the static solution and (b) relative to the upper bound, obtained from 50 random networks of 100 nodes and 4 sinks. Notched boxes indicate median and quartiles; whiskers indicate minimum and maximum values.

sensor networks. In addition, our power assignment algorithms require some degree of centralized control of the network. Either all the nodes need to be aware of the network's complete initial energy state, or they need to communicate with some central coordinating node. The problems of node mobility and distributed approximate optimization of the power assignment schedules remain to be studied.

7. REFERENCES

[1] M. Adler and C. Scheideler, "Efficient communication strategies for ad hoc wireless networks." *Theory Comput. Systems 33* (2000), 337–391.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and Approximation. Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag, Berlin, 1999.

[3] D. Bienstock and A. Bley, Capacitated Network Design with Multicast Commodities. Technical Report ZIB-Report 00-14, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2000.

[4] B. Bollobás, *Modern Graph Theory*. Springer-Verlag, New York NY, 1998.

[5] M. Čagalj, J.-P. Hubaux and C. Enz, "Minimum-energy broadcast in all-wireless networks: NP-completeness and distribution issues." *Proc. MOBICOM'02*, 172–182.

[6] I. Caragiannis, C. Kaklamanis and P. Kanellopoulos, "A logarithmic approximation algorithm for the minimum energy consumption broadcast subgraph problem." *Information Processing Letters 86* (2003), 149–154.

[7] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks." *Proc. INFOCOM'00*, 22–31. IEEE, New York NY, 2000.

[8] A. E. F. Clementi, P. Crescenzi, P. Penna, G. Rossi and P. Vocca, "On the complexity of computing minimum energy consumption broadcast subgraphs." *Proc. 18th Symp. on Theoretical Aspects of Computer Science (STACS'01)*, 121–131. Lecture Notes in Computer Science 2010. Springer-Verlag, Berlin, 2001.

[9] A. K. Das, R. J. Marks, M. El-Sharkawi, P. Arabshahi and A. Gray, "Minimum power broadcast trees for wireless networks: integer programming formulations." *Proc. INFOCOM'03*. IEEE, New York NY, 2003.

[10] A. Ephremides, "Energy concerns in wireless networks." *IEEE Wireless Communications*, August 2002, 48–59.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, New York NY, 1979.

[12] *IEEE Wireless Communications*, Special issue on energy-aware ad hoc wireless networks, August 2002.

[13] K. Jain, M. Mahdian and M. R. Salavatipour, "Packing Steiner Trees." *Proc. 14th ACM-SIAM Symposium on Discrete Algorithms (SODA 2003)*, 266–274. ACM, New York NY, 2003.

[14] L. M. Kirousis, E. Kranakis, D. Krizanc and A. Pelc, "Power consumption in packet radio networks." *Theoretical Computer Science 243* (2000), 289–305.

[15] Q. Li, J. Aslam and D. Rus, "Online power-aware routing in wireless ad-hoc networks." *Proc. MOBICOM'01*, 97–107. ACM, New York NY, 2001.

[16] W. Liang, "Constructing minimum-energy broadcast trees in wireless ad hoc networks." *Proc. MOBIHOC'02*, 112–122. ACM, New York NY, 2002.

[17] E. L. Lloyd, R. Liu, M. V. Marathe, R. Ramanathan and S. S. Ravi, "Algorithmic aspects of topology control problems for ad hoc networks." *Proc. MOBIHOC'02*, 123–134. ACM, New York NY, 2002.

[18] R. J. Marks II, A. K. Das, M. El-Sharkawi, P. Arabshahi and A. Gray, "Maximizing lifetime in an energy constrained wireless sensor array using team optimization of cooperating systems." *Proc. IEEE World Conference on Computational Intelligence 2002*. IEEE, New York NY, 2002.

[19] R. J. Marks II, A. K. Das, M. El-Sharkawi, P.

Arabshahi and A. Gray, "Minimum power broadcast trees for wireless networks: optimizing using the viability lemma." *Proc. IEEE Int. Symp. in Circuits and Systems 2002*, I:273–276. IEEE, New York NY, 2002.

[20] I. Papadimitriou and L. Georgiadis, "Energy-aware broadcasting in wireless networks." *Proc. WiOPT'03*, 267–277. INRIA Sophia-Antipolis, 2003.

[21] C. E. Perkins, *Ad Hoc Networking.* Addison-Wesley, Boston MA, 2001.

[22] H. J. Prömel and A. Steger, *The Steiner Tree Problem.* Vieweg, Braunschweig, 2002.

[23] R. Rajaraman, "Topology control and routing in ad hoc networks: A survey." *ACM SIGACT News 33:2* (June 2002), 60–73.

[24] R. Ramanathan and R. Rosales-Hain, "Topology control of multihop wireless networks using transmit power adjustment." *Proc. INFOCOM'00*, 404–413. IEEE, New York NY, 2000.

[25] T. S. Rappaport, *Wireless Communications: Principles & Practice.* Prentice Hall, Upper Saddle River NJ, 1996.

[26] V. Rodoplu and T. H. Meng, "Minimum energy mobile wireless networks." *IEEE J. Selected Areas in Communications 17* (1999), 1333–1344.

[27] S. Singh, M. Woo and C. S. Raghavendra, "Power-aware routing in mobile ad hoc networks." *Proc. MOBICOM'98*, 181–190. IEEE, New York NY, 1998.

[28] P.-J. Wan, G. Călinescu, X.-Y. Li and O. Frieder, "Minimum-energy broadcasting in static ad hoc wireless networks." *Wireless Networks 8* (2002), 607–617.

[29] J. E. Wieselthier, G. D. Nguyen and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks." *Proc. INFOCOM'00*, 585–594. IEEE, New York NY, 2000.

[30] J. E. Wieselthier, G. D. Nguyen and A. Ephremides, "Algorithms for energy-efficient multicasting in static ad hoc wireless networks." *Mobile Networks and Applications 6* (2001), 251–263.