Haanpää, Harri; Schumacher, André; Thaler, Thorn; Orponen, Pekka

Distributed computation of maximum lifetime spanning subgraphs in sensor networks

Published in:
The 3rd International Conference on Mobile Ad-Hoc and Sensor Networks (MSN'07, Beijing, China, December 2007)

DOI:
10.1007/978-3-540-77024-4_41

Published: 01/01/2007

Document Version
Early version, also known as pre-print

Please cite the original version:
Haanpää, H., Schumacher, A., Thaler, T., & Orponen, P. (2007). Distributed computation of maximum lifetime spanning subgraphs in sensor networks. In H. Zhang, S. Olariu, J. Cao, & D. B. Johnson (Eds.), *The 3rd International Conference on Mobile Ad-Hoc and Sensor Networks (MSN'07, Beijing, China, December 2007)* (pp. 445-456). https://doi.org/10.1007/978-3-540-77024-4_41

Distributed Computation of Maximum Lifetime Spanning Subgraphs in Sensor Networks *

Harri Haanpää, André Schumacher, Thorn Thaler, and Pekka Orponen

Lab. for Theoretical Computer Science, TKK – Helsinki University of Technology,
P.O. Box 5400, FI-02015 TKK, Finland
Harri.Haanpaa@tkk.fi, Andre.Schumacher@tkk.fi,
Thorn.Thaler@tkk.fi, Pekka.Orponen@tkk.fi

Abstract. We present a simple and efficient distributed method for determining the transmission power assignment that maximises the lifetime of a data-gathering wireless sensor network with stationary nodes and static power assignments. Our algorithm determines the transmission power level inducing the maximum-lifetime spanning subgraph of a network by means of a distributed breadth-first search for minmax-power communication paths, i.e. paths that connect a given reference node to each of the other nodes so that the maximum transmission power required on any link of the path is minimised. The performance of the resulting Maximum Lifetime Spanner (MLS) protocol is validated in a number of simulated networking scenarios. In particular, we study the performance of the protocol in terms of the number of required control messages, and compare it to the performance of a recently proposed Distributed Min-Max Tree (DMMT) algorithm. For all network scenarios we consider, MLS outperforms DMMT significantly. We also discuss bringing down the message complexity of our algorithm by initialising it with the Relative Neighbourhood Graph (RNG) of a transmission graph rather than the full graph, and present an efficient distributed method for reducing a given transmission graph to its RNG.

1 Introduction

Maximising the lifetime of a network, most commonly in terms of connectivity, is a key design goal in wireless sensor networks. Network longevity can be affected by many methods, ranging from hardware design to energy-aware routing [1]. We focus on *topology control*, specifically on assigning transmission power levels to the battery-operated nodes so that under a uniform traffic load the network remains connected for a maximum length of time [2]. We consider the case where the nodes are non-mobile and the power levels, once fixed, stay the same throughout the operating life of the network. An application scenario would be a sensor network whose main purpose is to provide sporadic status messages to a common sink node. Consider for example a sensor network that is deployed in a forest region to detect fire.

* Partially supported by the Academy of Finland under grant 209300 (ACSENT).

It is apparent that under our assumptions of stationary nodes and uniform traffic load, maximising the lifetime of a network is equivalent to finding the lowest possible transmission power levels for the nodes that suffice to make the network connected. This problem of minimising the maximum transmission power required to establish connectivity has been considered previously in the literature several times. One of the earliest papers on the topic is by Ramanathan and Rosales-Hain [3], who address the problem in the setting of maximising the lifetime of a single-session broadcast. They propose a centralised algorithm for finding the minimum maximum (minmax) transmission power level that maintains network connectivity, as well as two simple distributed heuristics that aim at achieving the same. Their distributed heuristics, however, are suboptimal and do not necessarily guarantee connectivity in all cases.

Kang and Poovendran [4] discuss several problems related to dynamic lifetime maximisation, such as non-uniform energy levels. They also emphasise the importance of considering the minmax energy metric rather than the more often addressed minimum total energy metric for maximising network lifetime. For a distributed implementation, Kang and Poovendran rely on distributed methods for constructing minimum spanning trees (MST), such as the algorithm of Gallager, Humblet and Spira [5]. These techniques are, however, rather involved, and we complement this work by suggesting a much simpler distributed method for constructing general spanning *subgraphs* with minmax edge costs.

The problem of minimising the *total*, as opposed to minmax, network transmission power required for connectivity has been studied extensively (cf. e.g. [2] and the references therein). Rodoplu and Meng [6] present a distributed algorithm for this problem that is based on the concept of *relay regions*: each node is aware of its own geographic location and the location of its neighbours. Based on a path-loss model, nodes can locally determine to which neighbour they should forward the message to minimise the total energy consumption. The algorithm proposed in [6] is optimal but requires extensive assumptions, such as the availability of location information and a specific path-loss model.

Wattenhofer, Li, Bahl, and Wang [7] propose a distributed algorithm for the same problem. Their algorithm, which relies on a geometric cone-based forwarding scheme, requires that nodes can measure exactly the direction of incoming radio transmissions (angle of arrival). It also makes further assumptions on geometric properties of the underlying graph model.

Furthermore, several researchers have proposed distributed algorithms that construct minimum spanning trees and can potentially be used for lifetime maximisation, e.g. the algorithm proposed in [5] or the self-stabilising algorithm by Gupta and Srimani [8]. An MST based problem formulation seems appropriate for minimising the total energy expenditure. However, the distributed construction of an MST is usually more involved than the distributed search for a spanning subgraph with minmax edge cost. In Section 3, we present a very simple and efficient distributed algorithm that finds such a spanning subgraph. For a discussion of the two different objectives, minimising total transmission power and minimising maximum transmission power, see e.g. [2, 4].

Our Maximum Lifetime Spanner (MLS) protocol is based on an approach similar to the distributed MST algorithm of Gupta and Srimani [8], viz. the construction of paths with minmax edge cost by breadth-first search similar to the asynchronous Bellman-Ford algorithm. However, as we do not consider the construction of an MST, we obtain several simplifications of the resulting scheme. Furthermore, we observe that before running the algorithm we can prune the network according to an algebraic formulation of relative neighbourhood graphs (RNG), for the purpose of avoiding traversal of redundant edges.

Recently Guo, Yang, and Leung [9] proposed a distributed algorithm DMMT (Distributed Min-Max Tree) for constructing minmax edge cost multicast trees, in the style of Prim's MST algorithm. Since their technique can easily be adapted also to sensor network lifetime maximisation, and seems to be the proposal in the literature closest to our MLS approach, we conducted an experimental comparison of the runtime behaviour of the two algorithms DMMT and MLS.

The rest of the paper is organised as follows. Section 2 gives a formal description of the lifetime maximisation problem. Section 3 describes our distributed method for finding a spanning subgraph with minmax transmission cost in a given network. Section 4 discusses a generalisation of RNGs and how they can be utilised for improvements of our algorithm. Section 5 presents a distributed algorithm for finding the RNG as an initial step of the algorithm proposed in Section 3. In Section 6 we evaluate our proposed Maximum Lifetime Spanner algorithm in terms of the number of required control messages, and compare it to the performance of the Distributed Min-Max Tree algorithm [9] proposed by Guo et al. using the `ns2` network simulator [10]. Section 7 presents our conclusions and outlines future research directions.

2 Lifetime Maximisation and Optimal *p*-Spanners

We model a sensor network as a graph $G(\tau) = (V, E(\tau))$, where the set of vertices V corresponds to the nodes of the network, $\tau : V \mapsto \mathbb{R}^+$ is a transmission power assignment, and the set $E(\tau)$ represents the directed links between nodes induced by a given transmission power assignment τ. We assume distinct node identifiers.

Each node has a finite energy budget that is consumed during the operation of the network. We assume that the initial value is the same for all nodes. We consider a scenario where the energy consumed by wireless communication dominates over energy consumed by computation or sensing. The minimum power a node u can use to maintain a link to a neighbouring node v is denoted by $\delta(u, v)$, where $\delta : V \times V \mapsto \mathbb{R}^+$ is the representative link cost function. We assume that the link costs are symmetric, i.e. $\delta(u, v) = \delta(v, u)$ for all $u, v \in V$; this is the case for example if the costs represent signal attenuation resulting from a deterministic path-loss model that only depends on the pairwise distance of nodes. In practice, one would expect a number of unidirectional communication links between the nodes and choose the maximum of the edge costs for δ such that bidirectional communication can be supported. We consider the notion of

lifetime that regards all nodes as equally important, so that the objective is to maximise the time span after which the first node runs out of energy [11].

The set $E(\tau)$ of edges in $G(\tau)$ is induced by the transmission power assignment τ by the rule that an edge (u, v) is an element of $E(\tau)$ if and only if the transmission power $\tau(u)$ at node u is at least $\delta(u, v)$. Each node has the same maximum transmission power $p_{\max}$ that must not be exceeded. We assume that the nodes can form a connected network by using full power, i.e., that $G(\tau_{\max})$ with $\tau_{\max}(u) = p_{\max}$ for all u is a connected graph.

We consider the problem of finding a static transmission power assignment $\tau : V \mapsto [0, p_{\max}]$ that maximises the lifetime of the network while retaining connectivity. In this context, the desired power assignment τ obviously induces a spanning subgraph with minmax edge cost α. Although this condition generally does not uniquely determine τ, choosing $\tau(u) = \alpha$ for all nodes u does not reduce the lifespan of the node that first runs out of energy. The power assignment τ is considered to be fixed after it has been once determined during the initial network setup. This property distinguishes this problem formulation from the computationally more complex problem of dynamically assigning transmission power levels [12].

Definition 1. *Given a set of nodes V and an edge cost function $\delta : V \times V \mapsto \mathbb{R}^+$, a graph $G = (V, E)$ is a p-spanner if G is connected and $\delta(u, v) \leq p$ for each edge $(u, v) \in E$. If no p'-spanner with $p' < p$ exists, then we say that the p-spanner is optimal.*

In other words, a p-spanner is a connected spanning graph for the nodes in V where no edge has cost greater than p. Note that for any network a $p_{\max}$-spanner exists exactly when the network can be connected by the nodes sending at full power. The lifetime maximisation problem is formulated as follows:

Definition 2. *Given a set V representing sensor network nodes and an edge cost function δ, find an optimal p-spanner $G = (V, E)$ for V and δ and determine a transmission power assignment $\tau : V \mapsto [0, p_{max}]$ such that $\max_{v \in V} \tau(v) \leq p$ and $\tau(u) \geq \delta(u, v)$ for each link $(u, v) \in E$.*

3 A Distributed Algorithm for Optimal p-Spanners

In the following, we describe a distributed algorithm that, given a graph G, finds a spanning subgraph of G with some minmax edge cost, i.e. an optimal p-spanner of G. Initially, we assume that each node v knows its neighbours in G and the cost between v and each of them. We assume that the links and costs are symmetric. Our algorithm finds a spanning subgraph – indeed, a spanning tree – with minmax edge cost as long as the original graph is connected. Following the reasoning presented in Section 2, the algorithm can then be used to determine the minmax transmission power that is required to maintain connectivity in a wireless sensor network. In this setting it would be run once during an initial setup phase of the network to distributively determine the transmission power

```
for node v with local variables α, f, α[·], status[·]
at start:
    α ← ∞; f ← undefined
    for u ∈ N(v):
        α[u] ← ∞; status[u] ← ready
    enter state SLEEP
in state SLEEP or state SEARCH:
    if (α′) with α′ < α is received from some node u then:
        if f is defined: send NAK(α) to f
        f ← u
        for w in N(v) \ {u}:
            if max(α′, δ(v, w)) < α[w]:
                send (max(α′, δ(v, w))) to w
                α[w] ← max(α′, δ(v, w)); status[w] ← wait
        enter state SEARCH
    if (α′) with α′ ≥ α is received from some node u then: send NAK(α′) to u
in state SEARCH:
    whenever status[w]=ready for all w ∈ N(v) \ {f}:
        send ACK(α) to f
        enter state SLEEP
    if ACK(α′) or NAK(α′) is received from u and α[u] = α′ then: status[u] ← ready
```

Algorithm 1. Distributed algorithm for finding an optimal p-spanner

level for each node. The graph $G(\tau_{\max})$ is then an obvious candidate for a graph to start from. Beneficial alternatives are discussed in Section 4 and 5.

Our Algorithm 1 for finding an optimal p-spanner is based on distributed breadth-first search similar to the asynchronous Bellman-Ford algorithm [13, Sec. 15.4]. However we use the properties of the minmax edge cost function to reduce the complexity of the search. First, a given reference node sends to each of its neighbours a message that contains the cost of the connecting edge. Upon first receiving the request, each node makes note of the node from which the message was received and rebroadcasts the request to its neighbours, updating the maximum edge cost α indicated in the request accordingly. Each node also remembers the best α sent to each neighbour. If a node that has already received and rebroadcast a request receives a request that indicates a better route from the reference node, it rebroadcasts the latter request to its neighbours if this leads to obtaining a route with a lower α, to those neighbours. In a typical data gathering scenario, the natural choice for the reference node is the node that collects the data.

Moreover, the nodes collect acknowledgements from their neighbours. When a node receives the request, it forwards it to its neighbours, and waits for each neighbour to either accept (ACK) or reject (NAK) it. When acknowledgements have been received from each neighbour, the node sends an ACK to the node from which it received the request. A NAK is sent if the node receiving the request already knows of a better path, or if a node learns of a better path while

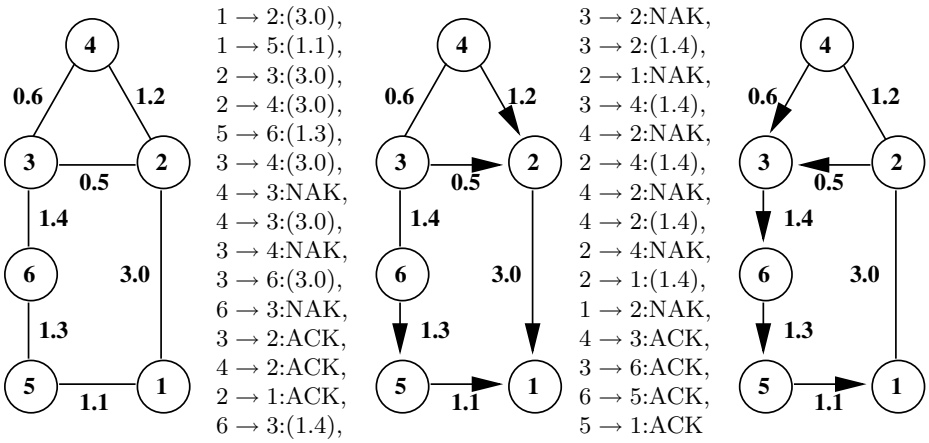


Fig. 1. Sample execution of Algorithm 1 from reference node 1; messages listed as *source→destination:message*. Initial state, intermediate state and final state with messages listed between states.

waiting for the acknowledgements from its neighbours. In this way, an ACK response means that the responding node has accepted the other node as its father in the tree being constructed, while a NAK signifies refusal. It can happen that a node will first respond with an ACK but later send a NAK; however, when the reference node has received acknowledgements from its neighbours, the algorithm has finished. To notify the remaining nodes about the termination of the algorithm, the reference node can then initiate a network-wide broadcast using the edges of the computed spanning tree. Each node v receiving this broadcast message can then decrease its transmission power $\tau(v)$ to the minimum power required to reach its father f_v and the neighbouring nodes that have chosen v to be their father. A sample run of Algorithm 1 is given in Figure 1.

In Algorithm 1, α is the current estimate of the minmax cost of a path from the reference node to each node v; and f is the node from which v has received the last accepted message. Initially, f is undefined and $\alpha = \infty$ for each v. The optimal p-spanner is defined by the f variables of each node after the algorithm has terminated.

To justify the algorithm, we firstly observe that it always terminates. Let Δ be the number of distinct edge costs in the graph; no node can learn of a new route with better α more than Δ times.

Secondly, at the end each node has a correct α_v: if from some node v there would exist a path of max cost $\alpha_0 < \alpha_v$ to the reference node, then on the path there is some edge of cost at most α_0 where exactly one endpoint would have a maximum edge cost estimate higher than α_0. This cannot happen, since the endpoint with cost at most α_0 should send a message along that edge. Thirdly, it cannot happen that a node would remain in the wait state, since its neighbours will respond to the queries either by an immediate NAK, if the cost was too large,

a delayed ACK once the neighbour has received responses from its children, or a delayed NAK in case the neighbour later learns of a lower max cost path.

To consider the communication complexity of the algorithm, observe that the number of distinct edge costs is bounded by $\Delta \leq |E|$ and with practical radio equipment, the number of distinct power levels is typically not large. In this regard the minmax edge cost spanner problem is different from finding minimum cost routes, where the number of routes with different total cost between two nodes can be exponential in the number of nodes [13, Sec. 15.4]. When a node learns of a better α, it will send a message to its neighbours, who will eventually answer with an ACK or a NAK. Since the requests sent by a node to its neighbour are in order of decreasing α, each of the $|E|$ edges participates in at most 2Δ updates, and the total communication complexity is $O(\Delta\,|E|)$.

4 Relative Neighbourhood Graphs

Algorithm 1 requires nodes to exchange messages with all neighbours. In a dense sensor network where the number of nodes within transmission range may be large, it is beneficial to limit the number of nodes that need to be contacted, while maintaining network connectivity at the same minmax transmission cost. For this purpose, we use *relative neighbourhood graphs* [14]. Relative neighbourhood graphs and related structures have been used for topology control [15,16], mostly in a geometric context, where nodes are placed in a plane and $\delta(u, v)$ depends only on the Euclidean distance between u and v. However, we only assume that path loss is symmetric, i.e. $\delta(u, v) = \delta(v, u)$. We will find, though, that when the nodes are placed in the Euclidean plane, our algorithm runs much faster.

Definition 3. *Given a graph $G = (V, E)$ and an edge cost function δ, the relative neighbourhood graph of G is the graph with vertex set V and edge set $\{\{u, v\} \mid \{u, v\} \in E, \nexists w$ s.t. $\{u, w\}, \{w, v\} \in E, \delta(u, w) < \delta(u, v), \delta(w, v) < \delta(u, v)\}$.*

In effect, the relative neighbourhood graph is obtained by deleting from each triangle in the original graph the edge with the highest cost. Such a generalisation of the concept of RNG has been already successfully applied to other problems, such as searching and broadcasting in peer-to-peer networks [17].

Proposition 1. *For any p, the* RNG *of G contains a p-spanner if G does.*

Proof. Consider an optimal p-spanner in the original graph. Order the k edges removed from the original graph in constructing the relative neighbourhood graph in increasing order of cost as $e_1, e_2, \ldots, e_k$. Let E_0 denote the edge set of the RNG, and let $E_i = E_{i-1} \cup \{e_i\}$ for $0 < i \leq k$. Suppose for contradiction that E_0 admits no p-spanner. Since E_k admits a p-spanner, there must be some least $0 < i^* \leq k$ such that E_{i^*} admits a p-spanner. By definition of the RNG, in E_{i^*-1} the endpoints of e_i are connected by a path of two edges shorter than e_i, so E_{i^*-1} also admits a p-spanner – a contradiction.

5 Distributed Algorithms for RNGs

In this section, we describe a distributed method for constructing RNGs. We do not assume that a node initially knows about the cost of the edges to its neighbours, but we assume that a node can estimate the strength of arriving radio signals, e.g. using Received Signal Strength Indication (RSSI) for a system with IEEE 802.11 network interfaces.

To construct the RNG, each node beacons at maximum power, sending out a message that contains its distinct node identifier. Nodes learn about their neighbours by receiving beaconing messages from them. They also estimate the path loss from the received signal strength. Path loss is used to estimate the δ-cost of a particular edge. We assume that path loss is symmetric; e.g. all path loss functions where the path loss depends only on the distance between the nodes fall into this category. In this manner all nodes can learn about their neighbours in $O(|V|)$ beaconed messages of $O(1)$ size.

After having learned about their neighbours, the nodes prune unnecessary edges from the graph formed by the nodes and radio links. To this end, the nodes again send beaconing messages. In addition to the identity of the beaconing node, this time the messages also contain the list of neighbours of the beaconing node, and the associated δ costs. If a node u learns, upon receiving a message from node v, that for some third node w it holds that $\delta(u, w) > \delta(u, v)$ and $\delta(u, w) > \delta(v, w)$, then u can determine that the edge (u, w) is not in the RNG, as per Definition 3. Thus the nodes can prune their neighbourhood so that only the RNG remains in $O(|V|)$ messages, the size of each of which is proportional to the number of neighbours the beaconing node has, and $O(|E|)$ in total.

Pruning the connection graph down to the RNG before running Algorithm 1 can give very considerable savings in complexity. With an arbitrary path loss function, the RNG can still contain $O(|V|^2)$ edges. However, when the nodes are in a plane and path loss is an increasing function of distance, the RNG is a subgraph of the Delaunay triangulation of the original graph and contains only $O(|V|)$ edges [14]. Thus in the Euclidean plane the communication complexity of the entire algorithm, including beaconing to determine neighbours, determining the RNG and computing an optimal p-spanner is $O(|E| + \Delta\,|V|)$.

6 Simulations

We experimentally validated Algorithm 1 and compared its runtime behaviour to the Distributed Min-Max Tree (DMMT) algorithm by Guo, Yang, and Leung [9] for a number of different scenarios using the `ns2` network simulator.

6.1 The DMMT Algorithm

Although DMMT was recently proposed for constructing maximum lifetime multicast trees in wireless ad hoc networks, it can be readily applied to solve the lifetime maximisation problem as formulated in Section 2. We focus on the

Table 1. Simulation parameters.

`ns2` version:	2.31	Square dimension:	919 m×919 m,1300 m×1300 m, 1592 m×1592 m,1838 m×1838 m
Transmission range:	250 m	Number of nodes:	50, 100, 150, 200
Interference range:	550 m	MAC protocol:	802.11 with `RTS/CTS`
Antenna type:	OmniAnt.	Propagation model:	TwoRayGround

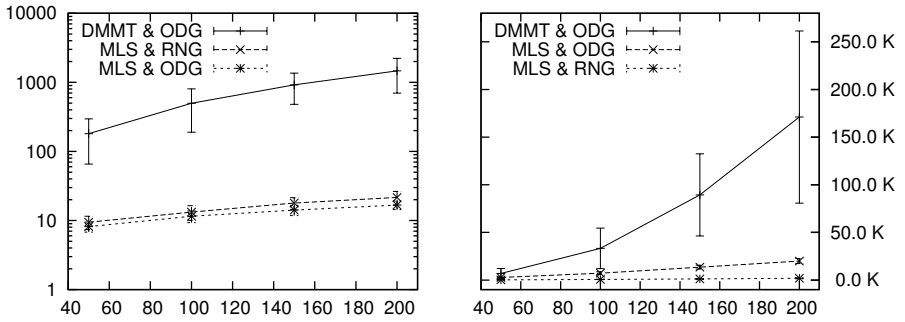
version of DMMT proposed in [9] for omnidirectional antennas. DMMT is based on Prim's well-known MST algorithm. The idea is to grow a subtree starting from the reference node, such that in each step the minimum cost edge is added that connects one node in the tree and another node not yet in the tree. After all nodes are added, the MST results.

The DMMT algorithm finds an optimal p-spanner (that is a tree) by adding an additional step to each iteration: after the minimum outgoing-edge-cost has been found, it is propagated to all tree nodes contained in a *join request* message. The tree nodes then forward this message to all adjacent nodes that are not yet in the tree using edges of cost at most the minimum outgoing edge-cost. Each node only forwards a join request once, and requests are identified by iteration counters. After a non-tree node is added via an edge incident to the tree node, the tree node becomes the *parent* of the added node which becomes a *child* of its parent. This is called the *growth phase*. After the growth phase has terminated, the next minimum outgoing edge-cost is determined in the subsequent *search phase*. In the search phase, each leaf node initiates a *join reply* message, which is forwarded along the tree towards the reference node. This message contains an estimate of the minimum outgoing-edge-cost in this iteration, which is updated as the message proceeds along the edges of the tree: each intermediate non-leaf node waits for all its children to send a *join reply* and then forwards a single *join reply* to its parent, that contains the minimum cost of the replies received from its children and the cost of its incident edges to non-tree nodes.

Guo et al. [9] have each node use timers to estimate the termination of the growth phase. However, to make DMMT more resilient against packet drops, we considered a more synchronised method where the source commands the nodes to switch from the growth to the search phase. The additional control messages required by our modification were not taken into account in comparison to MLS.

6.2 Experimental Evaluation of MLS

In evaluating MLS, we consider the number of control messages and the simulation time required. In our simulations, we use the *disk graph model*: the networks are created by randomly scattering nodes onto a square area with given dimensions, and connectivity is defined by the `ns2` default maximum transmission range. We discard disconnected graphs. Simulation parameters are summarised in Table 1. The dimensions of the square area were chosen to yield an expected density of one node per square of side length 130 m.



(a) Total running time

(b) Total number of messages

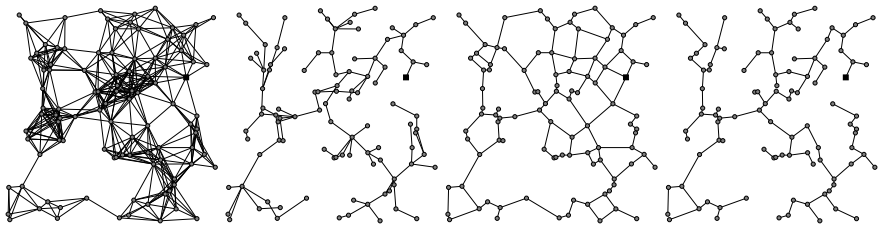
Fig. 2. Simulated running time (s) and number of messages required by MLS and DMMT on networks of varying size. Errorbars represent the standard deviation; for MLS results are shown for runs on the original disk graph (ODG) and on the RNG. Note the logarithmic scale in (a).

We implemented Algorithm 1 as a protocol for setting up a wireless network in `ns2`. We refer to this implementation as the Maximum Lifetime Spanner (MLS) protocol. MLS consists of Algorithm 1 and the method of setting the transmission power levels of each node after running Algorithm 1, as described in Section 3. At start, each node is assumed to know link cost to each of its neighbours. This input can be obtained by the beaconing algorithm in Section 5. For simplicity, we use Euclidean distance as link cost in the simulation. As the result of the algorithm, each node has a list of neighbours for whose reachability it is responsible, which the node then uses to set its transmission power.

Both MLS and DMMT give trees that form optimal p-spanners by setting transmission power levels as described above, for each of the network instances. In our `ns2` simulations, both algorithms converged despite a number of control packets being dropped by the MAC layer due to different reasons, such as network interference. Figure 2 shows the total (simulated) times required by the algorithms for convergence, where MLS is run on both the original disk graph (ODG) and on the RNG of the graph. As DMMT was insensitive to which input graph is used, for it only results on the original disk graph are given.

Our results indicate that MLS outperforms DMMT both in runtime and in the number of control messages transmitted, in particular when run on the RNG. MLS scales well with the number of nodes in the network, while DMMT shows a significant increase in the number of control messages required. However, the runtime of DMMT depends heavily on the values used to initialise the timers in the protocol, although the number of required control messages is unchanged.

Running MLS on the RNG instead of the original graph reduces the number of messages required, as indicated by Fig. 2, but it also removes paths with low



(a) Initial transmission graph

(b) Tree constructed by MLS on initial graph

(c) RNG of original graph

(d) Tree constructed by MLS on RNG

Fig. 3. Resulting minmax-power paths from the reference to the sensor nodes for a graph with 100 nodes; the remaining edges of the p-spanner induced by the corresponding transmission power are omitted for clarity.

minmax cost and a small hopcount. Indeed, the experiments show a slightly higher running time, as propagating ACKs and NAKs along the tree takes longer.

Figure 3 depicts one transmission graph instance, its RNG, and the tree that is constructed by MLS to calculate the transmission power levels for an optimal p-spanner for a network of 100 nodes.

7 Conclusions

We formulate the problem of lifetime maximisation in wireless sensor networks as a search for spanning subgraphs with minmax edge costs, which we call optimal p-spanners. We propose the MLS network protocol that determines the paths with minimum maximum edge cost. The algorithm is based on breadth-first search and is substantially simpler than methods relying on distributed minimum spanning tree algorithms. The `ns2` network simulator was used to compare the performance of MLS and DMMT in constructing minmax trees. In all scenarios considered MLS clearly outperforms DMMT in terms of number of control messages and execution time.

We also propose a distributed algorithm for extracting proximity graph structures. It uses beaconing to construct the RNG of the transmission graph of the network. We discuss the application of the algorithm to lifetime maximisation by integrating into a pre-processing stage before running the algorithm for finding optimal p-spanners. The resulting pruning of edges suggests significant gain in the efficiency of the original algorithm for dense networks.

To obtain meaningful results in practice, nodes must estimate the path loss for transmissions to their neighbours to a high accuracy. Furthermore, the path loss between neighbouring nodes has to be close to symmetric, as the edges in the spanning subgraph resulting from Algorithm 1 are likely to be used in the

opposite direction than they were added during the construction of the optimal p-spanner. A pairwise exchange of link cost information would remove the need for this assumption.

In the future we will integrate the RNG construction by beaconing into the MLS protocol. We also hope to consider distributed approximation algorithms for dynamic transmission power assignment. One extension that readily lends itself to the problem of dynamic power assignment is an iterative method based on single scaled subproblems for the static case.

References

1. Ephremides, A.: Energy concerns in wireless networks. IEEE Wireless Comm. **9**(4) (2002) 48–59
2. Lloyd, E.L., Liu, R., Marathe, M.V., Ramanathan, R., Ravi, S.: Algorithmic aspects of topology control problems for ad hoc networks. Mobile Networks and Appl. **10**(1-2) (2005) 19–34
3. Ramanathan, R., Hain, R.: Topology control of multihop wireless networks using transmit power adjustment. In: Proc. 19th Annual Joint Conf. IEEE Comp. and Comm. Societies. (2000) 404–413
4. Kang, I., Poovendran, R.: Maximizing network lifetime of broadcasting over wireless stationary ad hoc networks. Mobile Networks and Appl. **10**(6) (2005) 879–896
5. Gallager, R.G., Humblet, P.A., Spira, P.M.: A distributed algorithm for minimum-weight spanning trees. ACM Trans. on Programming Languages and Systems **5**(1) (1983) 66–77
6. Rodoplu, V., Meng, T.H.: Minimum energy mobile wireless networks. IEEE J. on Selected Areas in Comm. **17**(8) (1999) 1333–1344
7. Wattenhofer, R., Li, L., Bahl, P., Wang, Y.M.: Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In: Proc. 20th Annual Joint Conf. IEEE Comp. and Comm. Societies. (2001) 1388–1397
8. Gupta, S.K.S., Srimani, P.K.: Self-stabilizing multicast protocols for ad hoc networks. J. of Parallel and Distributed Computing **63**(1) (2003) 87–96
9. Guo, S., Yang, O.W.W., Leung, V.C.M.: Tree-based distributed multicast algorithms for directional communications and lifetime optimization in wireless ad hoc networks. EURASIP J. on Wireless Comm. and Networking **2007** (2007) Article ID 98938, 10 pages
10. McCanne, S., Floyd, S., Fall, K., Varadhan, K.: The network simulator `ns2` (1995) The VINT project, available for download at http://www.isi.edu/nsnam/ns/.
11. Chang, J.H., Tassiulas, L.: Energy conserving routing in wireless ad-hoc networks. In: Proc. 19th Annual Joint Conf. IEEE Comp. and Comm. Societies. (2000) 22–31
12. Floréen, P., Kaski, P., Kohonen, J., Orponen, P.: Lifetime maximization for multicasting in energy-constrained wireless networks. IEEE J. on Selected Areas in Comm. **23**(1) (2005) 117–126
13. Lynch, N.A.: Distributed Algorithms. Morgan Kaufmann, USA (1996)
14. Toussaint, G.T.: The relative neighbourhood graph of a finite planar set. Pattern Recognition **12** (1980) 261–268
15. Borbash, S., Jennings, E.: Distributed topology control algorithm for multihop wireless networks. In: Proc. 2002 Intl. Joint Conf. on Neural Networks. (2002)

16. Bhardwaj, M., Misra, S., Xue, G.: Distributed topology control in wireless ad hoc networks using β-skeletons. In: Workshop on High Performance Switching and Routing. (2005) 371– 375
17. Escalante, O., Pérez, T., Solano, J., Stojmenovic, I.: RNG-based searching and broadcasting algorithms over internet graphs and peer-to-peer computing systems. In: The 3rd ACS/IEEE Intl. Conf. on Computer Systems and Appl. (2005) 47–54