Data Transmission and Base-Station Placement for Optimizing the Lifetime of Wireless Sensor Networks*

Esther M. Arkin† Alon Efrat‡ Joseph S. B. Mitchell† Valentin Polishchuk§
Srinivasan Ramasubramanian‡ Swaminathan Sankararaman‡ Javad Taheri‡

Abstract

In this paper, we study the fundamental optimization problem in Wireless Sensor Networks of base-station positioning such that data from the sensors may be transmitted to it in an energy-efficient manner. We primarily consider the setting where a sensor transmits all of its data directly to the base-station or relays it via one other node. This setting provides two benefits: low duty-cycling due to limited synchronization requirements between nodes and low end-to-end delay due to the limited number of hops in the routes. Given the battery limitations of the sensor nodes, our objective is to maximize the network lifetime.

First, we present efficient algorithms for computing a transmission scheme for the sensors given a fixed base-station and show how to implement these in a distributed fashion with only a constant number of messages per sensor. Next, we show that the optimization problem for the setting where sensors may transmit data through more than 2 hops is NP-Hard. Finally, we present efficient algorithms for the problem of locating the base-station and simultaneously finding a transmission scheme.

We compare our algorithms with linear-programming based algorithms for more general settings through extensive simulations and outline the benefits of the different approaches.

1 Introduction

Recent advances in building powerful, highly integrated and energy-efficient electronic devices, coupled with emerging technologies in digital communication have enabled the creation of large-scale wireless sensor networks (WSNs) [3, 36]. WSNs have a vast number of applications, including military surveillance and tracking, environmental monitoring, human-centric applications and robotics (see, e.g., the survey of Arampatzis *et al.* [4]).

A WSN consists of a large number of sensor nodes deployed in a region to monitor and gather information, such as video, audio or temperature. The hardware of each node is typically limited in the resources it supports, with restricted processing and storage capabilities and limited power/energy (battery life). The information gathered by the sensor nodes is transmitted through wireless communication to a base-station which is linked to a central authority with more advanced processing capabilities. Thus, all data captured at a sensor node needs to be forwarded to the

*A Preliminary version of this paper appeared in [5].

†Stony Brook University.
‡The University of Arizona.
§University of Helsinki.

base-station. In related literature, the data forwarding scheme which establishes the connectivity pattern of the network is distinguished into two topologies: *Flat Network Topology*, in which all transmissions are homogeneous, i.e., each sensor node sends its data directly to the base-station, and *Multi-hop Network Topology*, in which transmissions are inhomogeneous, i.e., sensors can relay their data to the base-station through other sensor nodes. The cost of a transmission is impacted both by the data bit-rate and the distance of the transmission.

The major bottleneck in maintaining operation of the WSN for the longest possible time are the batteries of the sensors, which are depleted of their energy primarily due to the transmission of data rather than to sensing or computation [33, 21, 18, 25, 11]. Hence, locating the base-station and finding a transmission scheme for the sensor nodes such that the lifetime of the network is maximized is an important optimization problem. The lifetime of the network can be defined in various ways; in any case it reflects the length of the time period between the system start and some termination event. The latter event may be defined e.g., to be the moment of battery expiration of any one of the sensors, or of all of a specific set of sensors, or of a certain percentage of the sensors. An additional consideration is that the sensor nodes may be capable of recharging their batteries (at a certain rate, e.g., using a solar cell) which also may be taken into account.

Two factors that are central in the design and deployment of sensor networks are (1) low duty-cycle of the nodes to reduce power consumption and (2) low end-to-end delay of data delivery to maintain time-critical data. In order to decrease power consumption, it is of interest for the transmitters in the sensors to be turned off for long periods of time and activated only when transmissions occur. A schedule for switching the sensors from idle to active must optimize for minimizing the active times while also reducing the frequency of switching (which is expensive). In a multi-hop topology, the end-to-end delay is mainly impacted by the number of hops over which the data is relayed. Minimizing delay is directly conflicting with maintaining low duty cycles of nodes along the path taken by the data. Further, information loss when relaying which results in retransmission impacts the end-to-end delay as well. Hence, it is of interest to maintain as few relays as possible while reducing the power consumption of the nodes.

Clock Synchronization: A major consideration when designing a transmission schedule of nodes in a fully distributed network is clock synchronization between nodes [35, 32, 29, 15, 16, 34, 24]. This is necessary because for a transmission, the communication times for transmitter and receiver nodes needs to be synchronized in order to maintain a low duty-cycle. Since global synchronization of the clocks of all nodes requires extensive overhead, it is usually sufficient for a node to track deviations from its own clock, rather than to compare the clocks of all of its neighbors (the nodes with which it communicates). Thus, the active periods are adjusted to account for this divergence resulting in higher duty cycles. Further, the end-to-end delay is impacted by this divergence as well since the divergence is compounded at each hop.

The above discussion motivates our study. It is of interest to have infrequent transmissions to a limited number of nodes as this leads to multiple benefits including low duty-cycle, fewer clock synchronization operations and lower end-to-end delay. To maintain a low duty-cycle, transmissions must be of as infrequent as possible so as to increase idle times and reduce switching periods. In addition, the fewer the nodes to which transmissions occur, the lesser the resources spent in clock synchronization, if done locally. To reduce end-to-end delay, the data must travel as few hops as possible to reach the base-station. This not only reduces the delay of transmissions to reach the base-station but also reduces additional delay due to clock divergence of the nodes along

the transmission path. Thus, we are interested in topologies/transmission schemes which are not only shallow (route to base-station is short) but also have fewer transmissions (smaller number of necessary transmission links).

1.1 Contributions and Organization of the paper

In this paper, we investigate shallow tree topologies defining transmission schemes which maximize the *lifetime* of the network. The lifetime of the network is defined as the maximal time for which all sensor nodes are operating, or, in other words, as the time until at least one sensor exhausts its battery (as suggested by [13]). Shallow tree topologies exhibit both properties outlined above (hop count to base-station is low and have fewest number of transmissions) and compare these to other approaches. Here, the data transmission scheme/protocol is restricted to be a tree which is height-limited. We refer to a tree of height k as a k-tree. Restricting the transmission scheme to be a tree enforces each sensor to send *all* data (i.e., its own data plus data to be forwarded) to *one* recipient – either the base-station or another sensor, thus simplifying the clock synchronization process. Restricting the height reduces the route lengths to the base-station thus reducing end-to-end delay. We compare this to the case when the topology is not required to be a tree, i.e., a sensor can split the data between several other sensors with and without the restriction on the route lengths. In summary, this paper contains several contributions:

- In the context of 2-trees, we have two main contributions:
  - When the location of the base-station is given, we provide an exact algorithm to compute the optimal 2-tree (that which maximizes system lifetime) in time $O(n^{1.5}\log^3 n)$ and an approximate algorithm which computes a 2-tree which approximates the maximum system lifetime by a factor of at least $(1-\varepsilon)$ with running time $O(\frac{n}{\varepsilon^2}\log n \log\frac{1}{\varepsilon})$. Here ε is a tunable parameter in the range $(0,1)$ providing a tradeoff between accuracy and execution time. We also show how to implement this algorithm in a distributed fashion using only a constant (dependent on ε) number of messages per node.
  - Using the above algorithms, we show how to find the optimal location of the base-station with the associated 2-tree yielding maximum lifetime in time $O(n^5\log^3 n)$. Further, we show how we can find a location and associated 2-tree where the lifetime is approximated by a factor of $(1-\varepsilon)$ in $O(\frac{n^3}{\varepsilon^4}\log^3 n)$ time. Again, ε is a tunable parameter.
- We show that computing an optimal tree of 3 or more levels is NP-Hard.
- We formulate a linear programming problem (termed as CONSTRAINED-DAG) which finds an optimal topology while guaranteeing that each node's data reaches the base-station within two hops. In this formulation, data may be split and sent to multiple nodes. Through extensive simulations, we compare the lifetimes and parameters affecting it for the 2-trees computed by the above algorithms with this approach as well as with the case when the restriction on the number of hops is removed.

The organization of the paper is as follows. In Section 2, we define the settings of the network and formulate the problem. Section 3 deals with the concepts of *matchings* and $\vec{b}$*-matchings* which are central to our algorithms. Section 4 presents the proof that computing a 3-trees is NP-Hard and provides exact and approximate algorithms for computing 2-trees along with the distributed

implementation. Section 6 deals with the algorithms for base-station location while providing an associated 2-tree transmission scheme. The Constrained LP formulation and simulation results are presented in Sections 7 and 8 respectively, and finally, we present our conclusions.

1.2 Related Work

In the context of energy-conserving routing in sensor networks, Chang and Tassiulas [8] formulated the problem of routing information from a set of sources to a set of destinations with the objective of maximizing system lifetime (they defined the system lifetime to be the minimum lifetime over all nodes). They designed a linear program for the problem and provided several algorithms although no theoretical analysis of the running time of these was provided. Specifically for the case of routing to a single sink or base-station, the paper by Efrat *et al.* [13] addressed the case when the network is not constrained to be a tree and when there is no limit on the number of hops for the messages to reach the base-station. They split the problem into two parts: (i) find an optimal transmission scheme for a *given* base-station location, and (ii) find an optimal base-station location by using part (i) over a discrete set of locations. They showed that problem (i) can be expressed as a linear program (in the same way as [8]), which may be solved in polynomial time. For problem (ii), they proved that restricting the base-station to the locations of the sensor nodes achieves a constant-factor approximation for maximizing lifetime, and presented a $(1-\varepsilon)$-approximation by discretizing the search space into a limited number of points. Shi *et al.* [31] provided a general framework for base-station location problems with different objectives, and showed that their approach achieves a $(1-\varepsilon)$-approximation for the objective of maximizing lifetime. Our fundamental approach is most related to these works.

Buragohain *et al.* [7] also deal with the sub-problem of finding an optimal routing scheme to maximize the lifetime of the WSN when the base-station location is fixed. In their model the transmission and reception of 1 bit of data uses 1 unit of energy irrespective of the distance of transmission/reception. The number of hops in the transmission tree is not restricted. They prove that the problem is NP-hard.

In [27], Pan *et al.* considered the network to consist of clusters of nodes, with three types of nodes: *sensor nodes (SN)*, which generate and capture some real-time data, *application nodes (AN)*, which are responsible for collecting data from *SN*s, and a *base-station (BS)*, which collects data from all *AN*s. Thus, the network has a two-tier hierarchical topology, and the aim is to maximize the lifetime. Their conclusion is that the *AN*s are the most important nodes in prolonging the system lifetime. However, they do not obtain any theoretical bounds on the complexity of the problem.

Bogdanov *et al.* [6] tackled the case of multiple base-station positions. Two equivalent objectives were considered: (i) minimizing the recharging rate of the batteries of the sensors, while having a specified data generation rate, and (ii) maximizing the data generation rate, while respecting some fixed battery recharging rate. The aim was to find a *base-station location layout* for different numbers of base-stations, by considering a regular grid of unit cells in the WSN region.

Several works considered the model of a *mobile* base-station. Hou *et al.* [30] studied extending the life of a WSN by moving the base-station to different locations. In their model, as is typically the case, the base-station has an unlimited power supply. The idea is that since the sensors around the base-station consume more energy than others (because they work as relays for other sensors), one can move the base-station to different locations, to balance the battery consumption among all sensors. They considered two versions of the problem: (i) The number of base-station locations

is finite (C-MB), and (ii) The number of base-station locations is infinite (U-MB). They provide a polynomial-time algorithm for the C-MB problem and a $(1-\varepsilon)$-approximation algorithm for the U-MB based on discretizing the space into a set of finite locations. A survey of literature on base-station positioning can be found in [2].

2 Settings and Problem Formulation

A set $S \subset \mathbb{R}^2$ of n sensors is deployed in the plane. Each sensor is continuously generating data at a constant rate (normalized to 1), all of which needs to be transmitted to a base-station, denoted by $\mathbf{b}$ in $\mathbb{R}^2$. A sensor can either transmit its data directly to $\mathbf{b}$ or forward the data via other sensors. The battery capacity of each sensor is normalized to 1 unit.

For sensor networks sparsely deployed over a wide geographical area, the power consumption at a node is dominated by data communication components (transmission/reception) [19]. As in the book by Rappaport [28] and following the model of [13], we model the power dissipation at a receiver as follows:

$$p_r = c_r r_i, \tag{1}$$

where c_r is a constant and r_i is the incoming rate of received data. Additionally, the power required for transmission may be modeled as follows:

$$p_t(r) = [c_{t1} + c_{t2} D^{\alpha}] r_0, \tag{2}$$

where $p_t(r)$ is the power dissipated at a node transmitted at a transmitting node when the distance of transmission is D. Here, c_{t1} and c_{t2} are constants which are distance-independent, α is the path loss exponent with $2 \leq \alpha \leq 4$ [28] and r_0 is the output bit rate of the transmitter.

In order to simplify the model, we normalize all the constants to 1 such that the power at a node to transmit a distance D is simply D^{α} where $2 \leq \alpha \leq 4$. That is, if a node u sends its data to another node v, then u spends energy $|uv|^{\alpha}$, where $|uv|$ is the distance from u to v. Note that we implicitly assume that receiving the data comes at no cost; it is easy to see from the above equations that in our model, since the incoming bit rate is less than the outgoing bit rate (a node which relays data must transmit its own in addition to that it receives), one can think of the reception cost being included in that of sending. For simplicity, we further assume that the data rate and battery capacity are the same for all sensors.

Following [13], we are interested in maximizing the **lifetime** of the network, i.e., the maximal duration for which all nodes remain alive (are still powered).

2.1 Two-Level Tree Topologies

We restrict the sensors to transmit all data to one other node, i.e., each sensor cannot split the data between different nodes. In order to restrict to two-level trees, we make the additional assumption that any node which receives data for relaying must relay it to the base-station directly. We further assume that the data is "combinable", i.e., relaying sensors may combine the received data and combine it with data they generate in order to send it to the base-station.

As before, we denote the base-station by $\mathbf{b}$. The sensors $F \subset S$ that do not transmit their data directly to $\mathbf{b}$ are called *followers*; the sensors $L \subseteq S$ transmitting directly to $\mathbf{b}$ (while possibly forwarding data from followers) are called *leaders*. The links from followers to leaders and from leaders to the base-station form a tree of height 2, with $\mathbf{b}$ as the root, leaders on the first level,

Symbol	Definition
Common notation	
S	Set of sensor locations
$\mathbf{b}$	Base-station location
n	Number of sensors
τ	System Lifetime for a given base-station location and a transmission scheme
τ^*	Maximum System Lifetime
$\|uv\|$	Distance between two sensors u and v
α	Path Loss Exponent in Power Dissipation Model
ε	Tunable approximation parameter in the range $(0, 1)$
Two-Tree Algorithms	
L	The set of *leaders* (sensors transmitting directly to $\mathbf{b}$)
F	The set of *followers* (sensors whose data is relayed through leaders)
$\vec{b}(l)$	The capacity of a leader l (the maximum number of followers for which l may be a relay)
$d(l)$	The degree of a leader in the two-tree (the actual number of followers for which l is a relay in the two-tree plus the base-station)
b^*	The maximum capacity over all leaders
$\mathcal{L}$	The Layer Graph built by the HK algorithm
M	A partial $\vec{b}$-matching in the bipartite graph $G = (L \cup F, E)$
M^*	A maximum-cardinality $\vec{b}$-matching in $G = (L \cup F, E)$
A_τ	The arrangement of the orbits (circles around a sensor) for a given τ
LP and CLP	
x_{uv}	The variable in the LP for amount of data transmitted along edge from u to v
h_u	The average hopcount of data transmitted by node u

Table 1: Various notations used throughout the paper split into three categories: Common notation used throughout, notation used in algorithms for two-trees and notation used in the algorithms for general network topologies.

and followers on the second (leaf) level. The *degree* $\mathrm{d}(l)$ of a leader $l \in L$ is defined as its degree in the tree, i.e., as 1 plus the number of followers relaying their data via l. Let the maximum possible system lifetime in any of the settings be denoted by τ^*.

First, we consider the problem of a *given* base-station location. In this case, we split the problem of computing an optimal transmission scheme into two parts: (i) Given a target lifetime τ, find a feasible 2-tree (transmission scheme), that is, find a 2-tree such that the sensors can transmit data continuously for τ units of time, and (ii) Use the algorithm of part (i) as an oracle in finding the 2-tree which yields the maximum system lifetime τ^*.

Part (i) is equivalent to considering that the sensors have a finite amount of data collected which they must transmit at a transmission rate of 1 to the base-station (directly or indirectly). The objective is to find the 2-tree such that the number of sensors that can transmit their data is maximized. This is equivalent to finding a 2-tree such that the sensors may transmit continuously with a unit transmission rate for τ units of time. From now on, we deal with the latter version of the problem as it is makes comprehending the algorithm for part (ii) easier. In this version, a sensor s, transmitting its data and the data of k other sensors continuously to a location p at a unit rate, spends energy $(k+1)|sp|^\alpha$ *per unit time* (after normalization). Thus, transmitting for τ units of time requires energy $\tau(k+1)|sp|^\alpha$.

Formally, denoting by $L(f)$ the leader to which a follower $f \in F$ is transmitting, our problem is to check if the objective function value of the following problem is equal to n.

$$\begin{aligned} \max(|L| + |F|) \qquad \text{s.t.} \quad & \tau \mathrm{d}(l)|l\mathbf{b}|^\alpha \leq 1 \quad \forall l \in L \\ & \tau |fL(f)|^\alpha \leq 1 \quad \forall f \in F \end{aligned} \tag{3}$$

For the above problem, the tree-construction algorithm first chooses a set of followers F and then, chooses a leader $L(f)$ for each $f \in F$ such that the constraints are satisfied. If this is not possible, the problem instance is infeasible.

Extending to part (ii), the objective is to find a protocol maximizing the time τ^* before the first sensor runs out of battery while ensuring that all sensors send in their data for τ^* units of time. Formally, the problem is

$$\begin{aligned} \max \tau \qquad \text{s.t.} \quad & \tau \mathrm{d}(l)|l\mathbf{b}|^\alpha \leq 1 \quad \forall l \in L \\ & \tau |fL(f)|^\alpha \leq 1 \quad \forall f \in F \\ & F \cup L = S \end{aligned} \tag{4}$$

In this case, the algorithm works by comparing different values of τ and using the algorithm of part (i) as an oracle to find solution for each value of τ.

Finally, when the location of the base-station is not given, we find a location $\mathbf{b} \in \mathbb{R}^2$, i.e., we solve problem (4) when $\mathbf{b}$ is a variable.

Remark. *Although in our algorithms, we do not take into account the effects of interference on the transmissions, we note that it is possible to schedule the transmissions of the resultant schemes in an interference-free manner with only a constant-factor loss in lifetime (see [23, 22]). Essentially, we obtain a set of flows from sensors to the base-station consisting of one or two links each of which has a constant rate. Thus, the methods from [22, 23] can be applied to obtain an interference-free schedule.*

3 Preliminaries

Maximum $\vec{b}$-matchings and augmenting paths with respect to them are central to our algorithms, so we review them here.

Matchings. A *matching* in a bipartitegraph $G = (U \cup V, E)$ is a subset of edges such that every vertex is incident to at most one edge in the subset (Fig. 1). A maximum-cardinality matching in G can be found in $O(|E|\sqrt{|U \cup V|})$ time using the algorithm of Hopcroft and Karp [20] which follows the techniques of Dinic's algorithm for maximum flow [12]. Because our algorithms follow the classical schema of [20], we outline some details of the Hopcroft–Karp algorithm (abbreviated as HK) and its analysis.

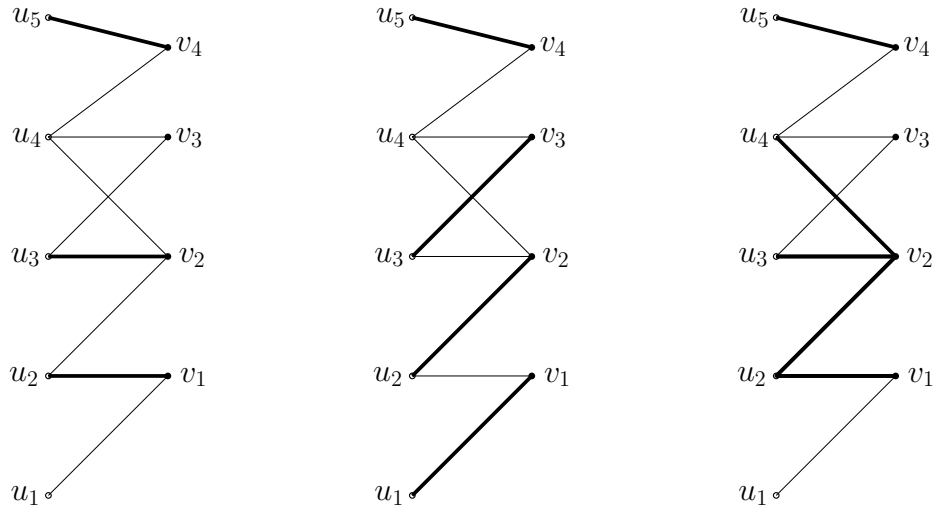


Figure 1: (Left) A matching consisting of edges u_1v_1, u_3v_2, u_5v_4. u_1, u_4 and v_3 are the exposed vertices. $u_1v_1u_2v_2u_3v_3$ is an augmenting path. (Middle) After flipping the edges along the augmenting path, a matching $u_1v_1, u_2v_2, u_3v_3, u_5v_4$ is obtained. (Right) A $\vec{b}$-matching for $\vec{b} = (0, 2, 1, 1, 1, 1, 3, 0, 1)$.

Let $M \subset E$ be a matching. An *exposed vertex* in M is a vertex which is not part of any edge in M. An *augmenting path* is an odd-length path that starts and ends with an exposed vertex and whose every other edge is not in M. An augmenting path exists if and only if M is not maximum. Once we have an augmenting path, we can increase the cardinality of M by 1 be making all matched edges in the path unmatched and the remaining edges matches (termed as "augmenting"). Algorithms to find maximum cardinality matchings repeatedly find augmenting paths and increase the matching at each step until no more augmenting paths are found. For more details on matchings, see [10, Chapter 26].

The main idea behind the HK algorithm is to compute all shortest augmenting paths simultaneously.

- We conduct a Breadth-First-Search (BFS) to obtain layers $L_1, \ldots, L_{2t}$ where $2t$ is the length of the shortest augmenting paths. The first layer consists of exposed vertices in U. L_{2i} contains all vertices of V connected to some vertex in L_{2i-1} and not seen in all previous layers. If L_{2i} contains exposed vertices, it is the last layer. Otherwise, L_{2i+1} consists of all

vertices in U matched to vertices in L_{2i}. Note that the odd layers contain vertices in U and even layers consist of vertices in V. The termination of the BFS means that there exists at least one augmenting path; the goal of the DFS is to find the path.

The layered graph $\mathcal{L}$ consists of vertices in $\cup_{i=1}^{2t} L_i$ and all edges between adjacent layers.

- Next, we conduct a Depth-First-Search (DFS) which goes back along the edges of $\mathcal{L}$, starting from an exposed vertex in L_{2t}. After an augmenting path is found, the path's vertices are removed, and a DFS is started from another exposed vertex in L_{2t}.

After the BFS+DFS phase, a set of *vertex-disjoint* augmenting paths is obtained. The edges along the augmenting paths are flipped (i.e., each edge in M is removed from M, while each edge not in M is added to M), resulting in a larger cardinality matching.

The analysis of the running time of the HK algorithm is based on the fact that after the kth BFS+DFS phase, all length-k augmenting paths are discovered. Moreover, at each phase at least one path is discovered and all remaining paths are *vertex-disjoint* [20, Corollary 4]; thus, the remaining number of phases is at most $|U \cup V|/k$. The running time follows from setting $k = \sqrt{|U \cup V|}$ and observing that one BFS+DFS phase takes linear time.

$\vec{b}$-Matchings. Let $\vec{b}$ be a vector of $|U \cup V|$ integers, with a component, $\vec{b}(v)$, associated with each vertex $v \in U \cup V$. A *$\vec{b}$-matching* is a subset $M \subseteq E$ of edges such that each vertex $v \in U \cup V$ is incident to at most $\vec{b}(v)$ edges of M. $\vec{b}$-matchings generalize matchings; a latter is a $\vec{b}$-matching with $\vec{b}(v) = 1$ for every $v \in U \cup V$.

The HK algorithm extends to finding a maximum-cardinality $\vec{b}$-matching: We extend the definition of an *exposed vertex* to a vertex whose degree in M is less than $\vec{b}(v)$. The layer graph $\mathcal{L}$ is built in the same manner except that multiple matched edges may be added when going from some layer L_{2i} to L_{2i+1}.

The correctness of the extension of the HK algorithm to $\vec{b}$-matching follows from the fact that, analogously to matchings, the symmetric difference between two $\vec{b}$-matchings is a set of alternating paths and cycles. However, the running time is worse in general as the vertex-disjointness of the augmenting paths after $\sqrt{n}$ phases is not guaranteed. In order to analyse the number of phases of BFS+DFS, we may consider each vertex $u \in U \cup V$ to be replicated $\vec{b}(v)$ times and consider a matching in the resulting graph. Thus, we have at most $O(\sqrt{b^*(|U \cup V|)})$ phases where $b^* = \max_{v \in U \cup V} \vec{b}(v)$ which gives us a running time of $O(|E|\sqrt{b^*(|U \cup V|)})$ since each BFS+DFS can still be implemented in $O(|E|)$ time.

Now, we show that as long as all vertices in either U or V are constrained to be matched to at most one edge, we may achieve the same running time as the HK algorithm (independent of b^*). This result proves useful in our setting.

Theorem 1. *If $\vec{b}(u) = 1$ for all $u \in U$, then, we may obtain a $\vec{b}$-matching in time $O(|E|\sqrt{|U \cup V|})$.*

Proof. Consider the graph $G' = (U \cup V', E)$ obtained by replicating each vertex $v \in V$ $\vec{b}(v)$ times and connecting each of these to all neighbors of v in G. Clearly, a matching in G' is a $\vec{b}$-matching in G. We know that after k layers, all further augmenting paths are of length at least k. Furthermore, these augmenting paths are vertex-disjoint in G', implying that in the original graph (before replication), they are vertex-disjoint with respect to vertices in U. Hence, the total number of paths is at most $|U \cup V|/k$. Thus, setting $k = \sqrt{|U \cup V|}$, just as in the original HK algorithm (and unlike general $\vec{b}$-matchings), the BFS+DFS phase is invoked at most $O(\sqrt{|U \cup V|})$ times. Clearly, each DFS takes $O(|U \cup V|)$ time leading to a total time of $O(|E|\sqrt{|U \cup V|})$. □

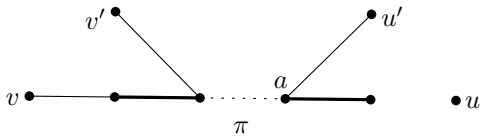


Figure 2: Edges of M are bold. $u' - a - u$ is an augmenting path in G with respect to M.

Maintaining a maximum $\vec{b}$-matching. Suppose we are given a maximum $\vec{b}$-matching M in G, and suppose that for some vertex $v \in V$, $\vec{b}(v)$ is increased by 1. We can update M efficiently to get the maximum $\vec{b}$-matching for the modified $\vec{b}$, by observing that an augmenting path π (if one exists) in G must start from v (or else M was not maximal). One BFS+DFS is enough to find π and to flip edges along it. The obtained matching is optimal in the modified instance, since there are no other augmenting paths, as we now argue. If an augmenting path $\pi' = v' - u'$ still exists, it must use some edges from π (or else M was not maximum). Let a be a vertex were π' meets with π. The subpath of π' to a, concatenated with π from a to one of π's endpoints, is an augmenting path in the original instance (G with M), contradicting maximality of M. Refer to Figure 2.

Similarly, suppose that $\vec{b}(v)$ is decreased by 1. As above, we can update M efficiently to get a maximum $\vec{b}$-matching for the modified $\vec{b}$. If M is still feasible, it is also optimal. Otherwise, remove any edge $uv \in M$ incident to v from the matching. An augmenting path π (if one exists) in G must, without loss of generality, start from u (or else M was not maximal). As above, one BFS+DFS phase finds the path and flips edges along it; also, as above, no other augmenting paths exist. Thus, we have

Lemma 3.1 (cf. [10, Problem 26-4])**.** *After one component, $\vec{b}(v)$, of $\vec{b}$ is changed by ± 1, the maximum cardinality $\vec{b}$-matching can be updated with one BFS+DFS phase.*

4 Computing a data transmission scheme

As described in Section 2, we split the problem of computing an optimal transmission scheme into two parts: (i) Given a target lifetime τ, find a feasible 2-tree (transmission scheme), that is, find a 2-tree such that the sensors can transmit data continuously for τ units of time, and (ii) Use the algorithm of part (i) as an oracle in finding the 2-tree which yields the maximum system lifetime τ^*.

4.1 Finding a feasible 2-Tree given a target lifetime τ

The problem of finding a feasible 2-tree for transmitting for τ units of time (Prob. (3)) can be reduced to bipartite $\vec{b}$-matching as follows. Splitting the sensors into followers and leaders is easy: If a sensor $s \in S$ is "far" from the base-station $\mathbf{b}$, precisely if $\tau |s\mathbf{b}|^\alpha > 1$, then s cannot transmit to $\mathbf{b}$, and, thus, $s \in F$. Moreover, by an exchange argument, in an optimal tree, any sensor $s \in S \setminus F$ can transmit directly to the base-station. Thus, $L = S \setminus F$.

Define the *capacity* of a leader $l \in L$ to be

$$\vec{b}(l) = \min\{n, \lfloor 1/(\tau |l\mathbf{b}|^\alpha) \rfloor\} - 1.$$

The capacity shows how many followers can forward their data via l. Set $\vec{b}(f) = 1$ for all $f \in F$. Let $G = (L \cup F, E)$ be the bipartite graph on the leaders and followers whose edges connect a

follower f to a leader l whenever f is close enough to l to transmit to it: $|fl|^\alpha \leq 1$. The crucial observation is the following.

Observation 4.1. *A transmission scheme maximizing the number of sensors that can transmit their data to the base-station can be obtained by computing maximum-cardinality $\vec{b}$-matching in G.*

Unlike general $\vec{b}$-matchings however, we may obtain a running time independent of the maximum capacity of the leaders. Observing that all followers can be matched to at most 1 leader, we obtain a running time of $O(n^{2.5})$ by Theorem 1.

We show how to improve the running time further to $O(n^{1.5} \log n)$ by exploiting the geometry of the problem. Our main ingredient in doing so is the following.

Lemma 4.2. *The BFS can be performed in $O(n \log n)$ time.*

Proof. Start the BFS from unmatched (exposed) vertices of F. Then the L-F step of the BFS is straightforward: just follow the matched edges from the reached vertices of L. We now describe how to perform the F-L step of the search. That is, we show how to do the following: Given a set of points $F' \subseteq F$ (exposed followers), find all leaders $L' \subseteq L$ such that a follower in F' can transmit to a leader in L'. This is equivalent to finding all leaders l such that the unit disk centered at l contains an exposed follower.

We discover all such disks one by one, spending $O(\log n)$ time per disk. Upon discovery, the disk is deleted, so that every disk is discovered only once. Thus, overall, discovering and deleting the disks takes $O(n \log n)$ time, since there are $O(n)$ disks. The discovery of the disks is done by scrolling through the exposed followers: For each exposed follower $f' \in F'$, in $O(\log n)$ time, we either discover (and delete) a disk that contains f' or report that no such disk exists (in which case we proceed to the next follower).

Using the data structure from the paper by Efrat *et al.* [14, Section 5.1], we can perform the discovery and deletion of disks in $O(\log n)$ time per disk. The structure has exactly the requisite properties: the capability to discover and delete one disk in $O(\log n)$ time. □

Combined with the fact that the BFS+DFS is invoked at most $O(\sqrt{n})$ times, this leads to the following lemma.

Lemma 4.3 (Exact Oracle). *Suppose that the location of the base-station and the target lifetime τ are given. In $O(n^{1.5} \log n)$ time, we can find a 2-tree under which all sensors transmit their data to the base-station for time τ, or determine that the maximum system lifetime τ^* is smaller than τ.*

Note that the number of edges in G can be quadratic in n. Nevertheless, we can find a maximum $\vec{b}$-matching in sub-quadratic time; of course, this is achieved by not building the graph explicitly.

4.1.1 An Approximate Oracle: Transmitting for $(1-\varepsilon)\tau$ units of time

In this section, we show how to design an approximate oracle, i.e., given a target lifetime τ, we find a transmission scheme where the associated system lifetime is at least $(1-\varepsilon)\tau$. Here, ε is a tunable parameter and provides a tradeoff between accuracy (how close $(1-\varepsilon)\tau$ is to τ) and the execution time.

Suppose that the range of transmission of the followers is increased from 1 to $(1+\delta)$ for some $\delta \in (0,1)$. Now, if there exists a transmission scheme with which it is possible to transmit for τ

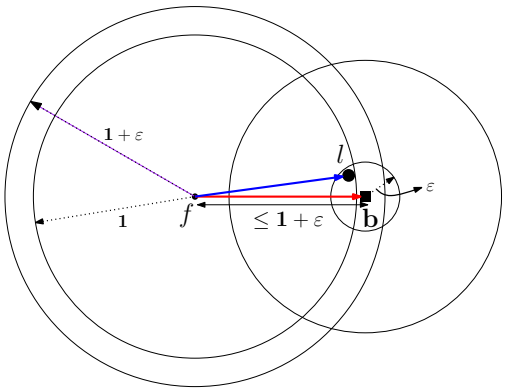


Figure 3: Leader l is within distance ε of $\mathbf{b}$ and hence, follower f transmitting to l (on the BLUE edge) can transmit directly to $\mathbf{b}$ (on the RED edge) in $G(1+\varepsilon)$. Thus, l does not need to forward any data.

units of time, we may more quickly be able to find a protocol with which we can transmit for atleast $(1-\varepsilon)\tau$ units of time. If the range of the followers is increased to $(1+\delta)$, we may think of this as the power spent per unit time as being at most $(1+\delta)^\alpha$. Thus, if a lifetime of τ was achievable when spending power 1 per unit time, we can achieve a lifetime of $(1-\varepsilon)\tau$ in this setting where $\varepsilon = O(\delta)$.

Formally, let $G(1)$ denote the bipartite graph when the range of followers is 1; define $G(1+\varepsilon)$ similarly. Let M^* be the maximum matching in $G(1)$, and let M be a matching in $G(1+\varepsilon)$ such that $|M| < |M^*|$. Clearly, M is not maximum in $G(1+\varepsilon)$, and thus there exists an augmenting path π with respect to it. We now argue that there actually exists a *short* augmenting path π'.

Lemma 4.4. *In $G(1+\varepsilon)$ there exists an augmenting path with respect to M of length $O(\varepsilon^{-2})$.*

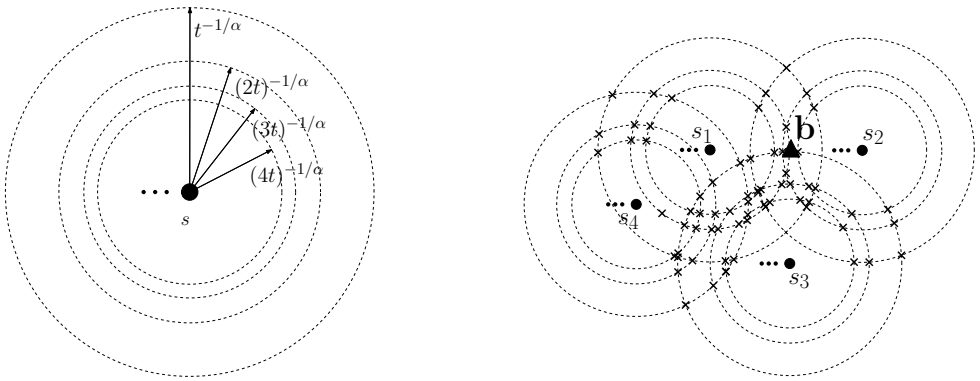
Proof. Let $\pi = (f_1, l_1, f_2, l_2, \ldots, f_k, l_k)$ be an augmenting path with respect to M in $G(1+\varepsilon)$, where f_i is a follower and l_i is a leader. We argue that there is an augmenting path, $\pi' = (f'_1, l'_1, f'_2, l'_2, \ldots, f'_{k'}, l'_{k'})$, with each $f'_i \in \{f_1, f_2, \ldots, f_k\}$ and each $l'_i \in \{l_1, l_2, \ldots, l_k\}$, for $i = 1 \ldots, k'$.

First, we set $f'_1 = f_1$. For each $i = 1, \ldots, k$ let $l'(f_i)$ be the last leader in the ordered sequence $(l_1, \ldots, l_k)$ such that $|f_i l'(f_i)| \leq 1+\varepsilon$. Now, we set $l'_1 = l(f'_1)$ and set f'_2 to be the follower immediately after l'_1 in the sequence π (i.e., if $l'_1 = l_{j_1}$, then $f'_2 = f_{j_1+1}$). Continuing in this way, we set $l'_i = l(f'_i)$ and f'_{i+1} to be the follower immediately after l'_i in the sequence π (i.e., if $l'_i = l_{j_i}$, then $f'_{i+1} = f_{j_i+1}$).

Now, we note by the choice of $l'_i = l_{j_i}$ that there are no leaders in the set $\{l_{j_1+1}, \ldots, l_k\}$ within the disk, $B_\varepsilon(l_1)$, of radius ε centered at l_1, since such a leader would be later in the sequence π than $l'_1 = l_{j_1}$, yet within distance $1+\varepsilon$ of $f'_1 = f_1$, contradicting the choice of $l'_1 = l(f'_1)$. Similarly, there are no leaders in the set $\{l_{j_2+1}, \ldots, l_k\}$ within the disk $B_\varepsilon(l_{j_1+1})$, and, more generally, no leaders in the set $\{l_{j_{i+1}+1}, \ldots, l_k\}$ within the disk $B_\varepsilon(l_{j_i+1})$, for $i = 1, 2, \ldots, k'-1$.

The k' disks $B_\varepsilon(l_1), B_\varepsilon(l_{j_1+1}), B_\varepsilon(l_{j_2+1}), \ldots, B_\varepsilon(l_{j_{k'-1}+1})$ therefore have the property that their centerpoints are pairwise separated by at least ε (no centerpoint lies in any other disk), implying, by a packing (area) argument, that $k' = O(\varepsilon^{-2})$, as claimed. □

Lemma 4.4, combined with Lemma 4.2, gives the following oracle:



(a) *Orbits of a sensor s*

(b) *Arrangement A_τ of the orbits of all sensors*

Figure 4: The circles of s and the circles arrangement.

Lemma 4.5 (**Approximate Oracle**). *Suppose that the location of the base-station and the target lifetime τ are given. If the maximum system lifetime τ^* is at least τ, we can find a 2-tree under which the system lifetime is at least $(1-\varepsilon)\tau$ in $O(\frac{n}{\varepsilon^2}\log n)$ time.*

Now, we note that, for matching M, leaders located inside the disk, $B_\varepsilon(\mathbf{b})$ of radius ε centered at $\mathbf{b}$ need not forward any data; any followers that were, in $G(1)$, to forward their data via leaders within $B_\varepsilon(\mathbf{b})$ can instead forward, in $G(1+\varepsilon)$, data directly to $\mathbf{b}$, since $\mathbf{b}$ lies within distance $(1+\varepsilon)$ of such followers (see Figure 3). Thus, we obtain the following lemma:

Lemma 4.6. *The maximum capacity of leaders in $G(1+\varepsilon)$ may be constrained to $1/\varepsilon^2$.*

This lemma proves very useful later on when designing a distributed algorithm and when optimizing the base-station location.

4.2 Finding the Maximum System Lifetime

Assuming we know τ^*, we may define the *orbits* of a sensor s to be the circles of radii $(\tau^*)^{-1/\alpha}, (2\tau^*)^{-1/\alpha}, \ldots, ((n-1)\tau^*)^{-1/\alpha}$ centered at s (Figure 4a). It is easy to see that there must exist two sensors i, j such that j lies on an orbit of i; otherwise, a small increase of τ^* does not change the feasibility of the transmission tree (while improving the objective function). Thus, τ^* belongs to the set of $O(n^3)$ "critical" candidate values at which a sensor j lies on an orbit of sensor i. After computing the critical values, we can sort them and do a binary search to determine the largest feasible critical value. During the search, a candidate value is tested using the exact oracle of Lemma 4.3. Since we make $O(\log n)$ queries, sorting the critical value is the bottleneck in the algorithm; its overall running time is $O(n^3 \log n)$ time.

To improve the running time, instead of checking *every* candidate value, we can use the following approach, based on the parametric search method of Megiddo [26]. Consider the arrangement A_τ of the circles of sensors (Figure 4b). The main idea here is to imagine the process of increasing τ from 0 until it exceeds the optimum, while keeping track of the vertices of A_τ. Special events occur when two vertices coincide and switch their order along the boundary of a third disk; this happens at the critical events. So we can use the parallel sorting networks of AKS [1], with the improvement described by Cole [9], as the generic algorithm. We refer the reader to [26] for further reading. In

our case, the sorting scheme attempts to find the order of all the vertices of the arrangement at τ^* along each of the orbits. There are $O(n^2)$ orbits giving rise to $O(n^4)$ intersection points (vertices of the arrangement). So the total number of values to be sorted is $O(n^4)$. Thus, the number of steps in the parametric search is only $O(\log n^4) = O(\log n)$ candidate values for τ. Each test is done by the oracle from Lemma 4.3.

Theorem 2. *Maximum system lifetime can be found in $O(n^{1.5} \log^2 n)$ time.*

4.2.1 Approximating maximum lifetime

One feasible solution is to set $\tau = 1/|s\mathbf{b}|^\alpha$ and have all sensors transmit directly to $\mathbf{b}$; here s is the sensor farthest from $\mathbf{b}$. This, in fact, is a $1/2^\alpha$-approximation because either s or its leader has to transmit to distance at least $|s\mathbf{b}|/2$.

Proposition 4.7. *A $1/2^\alpha$-approximation of the maximum system lifetime can be found in linear time.*

The approximation factor can be increased to $(1-\varepsilon)$ by a binary search: Suppose that we have an interval $[a, b]$ in which the maximum system lifetime τ is known to lie (this means that we have a b/a-approximation). Using the approximate oracle of Lemma 4.5, we can query whether the time $\tau' = \sqrt{ab}$ is feasible. Thus, we will have a $\sqrt{b/a}$-approximation. Since we start from $b/a = 2^\alpha$, making $\log(\alpha \ln 2 \,/\, \varepsilon)$ query steps is enough to improve the approximation factor from $1/2^\alpha$ to $(1-\varepsilon)$. Thus,

Theorem 3. *Maximum system lifetime can be $(1-\varepsilon)$-approximated in $O(\frac{n}{\varepsilon^2} \log n \log \frac{1}{\varepsilon})$ time.*

Note the close similarity between Theorem 3 and Theorem 2; the binary search in the former is replaced by the parametric search in the latter, and the approximate solution in the former is upgraded to an exact solution in the latter. This is the general feature of the parametric search.

4.3 Finding an optimal 3-level tree is NP-hard

In the previous section we considered finding optimal trees with at most 2 levels. We now justify our focus on only 2-trees by proving that if we are allowed to have 3 or more levels, our problems become NP-hard. We show that the problem is hard even in the case of deciding if the sensors can transmit for unit time (i.e., there is no issue of maximizing system lifetime). This implies that the problem is NP-hard in the case of continuous data transmission as well. As before, the only rule is that a sensor s can transmit to a point p if $\mathrm{d}(s)|sp|^\alpha \leq 1$, where $\mathrm{d}(s)$ is the total number of sensors (including s itself) whose data is forwarded by s.

Our decision problem is: Given $S, \mathbf{b}$, does there exist a tree (with possibly arbitrarily many levels) enabling all sensors in S to transmit their data for 1 unit of time to $\mathbf{b}$? To show the hardness, we reduce from 3PARTITION: given a set of integers $a_1 \ldots, a_{3n}$, decide if it is possible to split them into n groups of 3 so that the sum of the integers in every group is the same. The problem is hard even if each integer is between $B/4$ and $B/2$, and $B = O(1)$; here $B = \sum a_i/n$ [17]. Given such an instance of 3PARTITION, we create an instance $S, \mathbf{b}$ of the forwarding problem as follows (Figure 5).

The base-station $\mathbf{b}$ is at the origin. The sensors are of 3 types: n *cluster-heads*, $3n$ *elements*, and B *atoms*. For an integer m, let $d_m = m^{-1/\alpha}$ denote the distance for which the information of

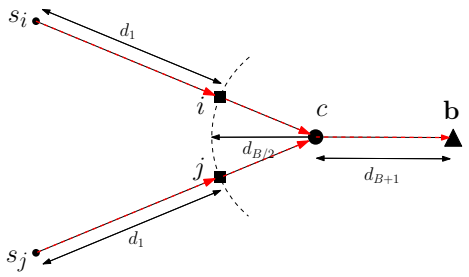


Figure 5: c is a cluster-head; i, j are two elements. s_i, s_j are atoms associated with i and j.

m sensors can be forwarded. We place all cluster-heads on the x-axis at distance d_{B+1} to the left of $\mathbf{b}$.

The elements correspond to the integers in the 3PARTITION. The $3n$ elements are placed on the semicircle of radius $d_{B/2}$ centered at the cluster-heads; the whole semicircle is to the left of the cluster-heads (we elaborate on the exact placement of the elements below). For each element i, we place $a_i - 1$ atoms at the point at distance d_1 from i on the ray from the cluster-heads through i.

If the 3PARTITION instance is feasible, the sensors can transmit the data as follows: The atoms transmit to their elements. The elements are split into n groups of 3 and each group transmits to one of the n cluster-heads. Finally, the cluster-heads transmit to $\mathbf{b}$.

It is easy to see that the above tree is, in fact, the only possible solution, for appropriately chosen locations of the elements (described below). First, the data from an atom can reach $\mathbf{b}$ only via its element; otherwise, if an atom s transmits to an atom s', the latter will not be able to forward its and s's data via its element.

Each element i has to transmit a_i units of data ($a_i - 1$ from its atoms plus its own unit). Now, in our construction above, we are free to place the elements *anywhere* on the semicircle. We choose to place them very close to the x-axis, so that the distance from any element to $\mathbf{b}$ is (almost) $d_{B+1} + d_{B/2}$. Because $a_i > B/4$, no element can transmit directly to $\mathbf{b}$ as soon as:

$$d_{B+1} + d_{B/2} > d_{B/4}. \tag{5}$$

Assuming (5) holds, the data from any element cannot reach the base-station directly, and has to go via a cluster-head. No cluster-head can transmit more than $B + 1$ units of data, and hence elements have to be split evenly among the cluster-heads.

Theorem 4. *Deciding whether there exists a protocol (with possibly arbitrarily many levels) enabling all sensors to transmit their data to the base-station, is NP-hard.*

Proof. We only have to show that (5) holds. This follows from simple arithmetic:

$$\begin{aligned}
(5) \Leftrightarrow \quad & (B+1)^{-\frac{1}{\alpha}} + \left(\frac{B}{2}\right)^{-\frac{1}{\alpha}} > \left(\frac{B}{4}\right)^{-\frac{1}{\alpha}} \\
\overset{B\geq 1}{\Longleftarrow} \quad & (2B)^{-\frac{1}{\alpha}} + \left(\frac{B}{2}\right)^{-\frac{1}{\alpha}} > \left(\frac{B}{4}\right)^{-\frac{1}{\alpha}} \\
\Leftrightarrow \quad & 2^{\frac{3}{\alpha}} - 2^{\frac{2}{\alpha}} - 1 < 0 \qquad \Leftarrow \quad \alpha \geq 2
\end{aligned}$$

□

5 A Practical Distributed Implementation

In this section, we show how to find a transmission scheme approximating the optimal lifetime in a distributed manner. The resulting transmission scheme provides a lifetime which is at least $(1-\varepsilon)\tau^*$ where τ^* is the maximum system lifetime and the distributed algorithm takes $O(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon})$ rounds with a message complexity of $O(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon})$ per node.

Our scheme requires that all sensors know their location and that we have global clock synchronization in the network when the algorithm is executed. Since our algorithm executes in constant number of rounds (only dependent on ε and not the number of sensor nodes), the clock skew can be expected to be insignificant during this period and hence, an initial synchronization is sufficient. Finally, we assume that the interference is not present during this time.

Similar to the description in Section 4, we first give a distributed algorithm to find the 2-Tree given a target lifetime τ and, as in Section 4.7, we execute a binary search to approximate the maximum lifetime. Recall that, to find a $\vec{b}$-matching for the centralized algorithm, we execute a set of phases each involving one Breadth-First-Search (BFS) followed by a Depth-First-Search (DFS). Lemma 4.2 shows how to perform the BFS step by using the data structure from [14, Section 5.1]. For the distributed approach, we essentially replace this data structure with a broadcasting scheme.

We now describe the 3 phases of the scheme:

- **Role Assignment:** This is the initialization phase of the algorithm. Here, the base-station **b** broadcasts a message to all nodes which are within distance 1 from it. All nodes which hear the message mark themselves as *leaders* and assign themselves the appropriate capacities depending on the distance from the base-station. Next, the leaders broadcast a message to all their neighbors (followers within distance 1) and all followers which hear this message add the appropriate association.

- **Layering:** In this phase, a layered graph is built implicitly. In the first time step, each exposed vertex, i.e., a follower which is marked as exposed broadcasts a message to all its neighboring leaders. In the next time step, the leaders broadcast messages to all their matched followers and so on till all the layers have been explored. Thus, the data structure from Lemma 4.2 is replaced by the distributed message broadcast. Finally, when there is an exposed vertex which receives a message adding it to a layer, the vertex broadcasts a message to all the nodes instructing them to stop the layer building process.

- **Augmenting:** In the last phase, the augmenting paths are built by sending messages along the paths backwards from the exposed leader which received the last message.

We start with the *Role Assignment* phase. Then we iterate between the *Layering* and *Augmenting* phases until each node is assigned a node to transmit its data to. The 2-Tree is not explicitly stored but the parent pointers in the tree are stored at all the nodes.

Analysing the algorithm, it is clear that the role assignment and Layering phases take only $O(1)$ messages per node. Only the augmenting phase is more expensive. Here, each leader v might have to send $\vec{b}(v)$ messages per layering phase. Thus the total message complexity per node per value of τ to find the feasible 2-Tree will be $O(b^*\sqrt{n}) = O(n\sqrt{n})$ in the general case.

This can be drastically reduced if we are interested in approximating the optimal lifetime. Note that, due to Lemma 4.6, each leader will have to send only $O(\varepsilon^{-2})$ messages per augmenting phase. Also, due to Lemma 4.4, we may restrict the number of layering phases to $O(\varepsilon^{-2})$. So, the

total number of messages per node is only $O(\varepsilon^{-4})$. Further, the total number of rounds for each augmenting phase is also only $O(\varepsilon^{-2})$ since the number of layers is $O(\varepsilon^{-2})$ by Lemma 4.4.

The total number of values of τ compared during the course of the algorithm is $O(\log \frac{1}{\varepsilon})$ according to Theorem 3. Thus, we get the following theorem:

Theorem 5. *The 2-Tree which $(1-\varepsilon)$-approximates the maximum lifetime can be found in a distributed fashion using $O(\frac{1}{\varepsilon^4}\log\frac{1}{\varepsilon})$ messages per node in $O(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon})$ rounds.*

6 Computing an Optimal Base-Station Location

We now assume that the location of the base-station is not given, and consider the problem of finding the base-station location and the transmission tree that together maximize the system lifetime. That is, the input to the problem is only the set of sensors; the sensors continuously generate data over time at uniform rate. The output is the location $\mathbf{b}$ for the base-station and a 2-level transmission tree such that the sensors can forward data to $\mathbf{b}$ for the longest possible time τ^*.

6.1 Exact solution

The problem can be solved by observing that it is enough to choose the best location for $\mathbf{b}$ from only a polynomial number of candidates. First of all, the base-station is at a vertex of A_τ; otherwise it can be moved while not intersecting any circle, thus ensuring that every leader can still transmit the necessary data to $\mathbf{b}$. Next observe, that as τ grows, all circles defining A_τ shrink, and $\mathbf{b}$ (as well as all other vertices of A_τ) moves along a curve defined by $\tau\dot{\mathrm{p}}|p\mathbf{b}|^\alpha = \tau\dot{\mathrm{q}}|q\mathbf{b}|^\alpha$ for some $p, q \in S$. If $\tau = \tau^*$, it also should be "locally maximal" in the sense that such a movement would make $\mathbf{b}$ infeasible. That is, either $\mathbf{b}$ is a point of tangency of two circles from A_τ, or is a point of intersection of three circles. In the first case, we have

$$\tau\dot{\mathrm{p}}|p\mathbf{b}|^\alpha = \tau\dot{\mathrm{q}}|p\mathbf{b}|^\alpha \quad |p\mathbf{b}| + |q\mathbf{b}| = |pq|, \tag{6}$$

and in the second,

$$\tau\dot{\mathrm{p}}|p\mathbf{b}|^\alpha = \tau\dot{\mathrm{q}}|q\mathbf{b}|^\alpha = \tau\dot{\mathrm{s}}|s\mathbf{b}|^\alpha, \tag{7}$$

for some $p, q, s \in S$.

There are $O(n^3)$ triples of points p, q, s, and $O(n^3)$ values for the degrees $\dot{\mathrm{p}}, \dot{\mathrm{q}}, \dot{\mathrm{s}}$; thus in $O(n^6)$ time we can build all locations for $\mathbf{b}$ that satisfy the above equations. For each location, the equations define a value of τ, and we can test whether all sensors can transmit for time τ by the oracle from Lemma 4.3. Thus,

Proposition 6.1. *The location for the base-station and the 2-level forwarding tree enabling transmission of data by all sensors for maximum time can be found in $O(n^{7.5}\log n)$ time.*

To improve the running time we use parametric search [26] with the following oracle:

Lemma 6.2. *Suppose that the the target lifetime τ is given. In $O(n^5 \log n)$ time, one can find a location for the base-station and a forwarding protocol under which all sensors transmit their data to the base-station for time τ, or determine that the maximum system lifetime is smaller than τ.*

Proof. Build the arrangement A_τ. As argued above, $\mathbf{b}$ may be taken to be one of the $O(n^4)$ vertices of A_τ. For each vertex we can test if τ is feasible by the oracle from Lemma 4.3, obtaining an $O(n^{5.5} \log n)$-time algorithm.

To decrease the runtime, note that the problem does not have to be solved "from scratch" at every vertex of A_τ. Instead, start by computing the optimal protocol T_p for $\mathbf{b}$ located at an arbitrary vertex p of A_τ. Move $\mathbf{b}$ to an adjacent vertex, q. Suppose that p is the intersection of circle centered at some sensors i, j, and that q is the intersections of circles centered at i, k (possibly, $k = j$). As $\mathbf{b}$ is moved from p to q, the only changes in the graph $G = (F \cup L, E)$ concern the sensors j and k: they change their capacity by 1 (including the possibility of switching between F and L). Thus, by Lemma 3.1, the optimal protocol T_q can be obtained from T_p by one BFS+DFS phase, which takes $O(n \log n)$ time by Lemma 4.2. Hence, in overall $O(|A_\tau|)$ steps we can walk over all vertices of A_τ, moving between adjacent vertices, spending time $O(n \log n)$ per vertex to update the optimal protocol. □

As argued above, there are $O(n^4)$ critical values for τ that are sufficient to test. Instead of testing each value (as done in Proposition 6.1), we can use the same approach as described in Section 4.2, based on the parametric search method of Megiddo [26] as in Section 4.2. Overall, the parametric search allows us to test only $O(\log n^4) = O(\log n)$ candidate values for τ. Each test is performed by the oracle from Lemma 6.2.

Theorem 6. *The location for the base-station and the 2-level forwarding tree enabling transmission of data by all sensors for maximum time can be found in $O(n^5 \log^2 n)$ time.*

Remark 6.3. *It is reasonable to assume that a leader would relay data only from a small number $K = O(1)$ of sensors. In addition, even if this is not the case and a sensor s does relay data from more than K other sensors, the boundaries of the disks centered at s corresponding to larger values of K tend to cluster very close to each other around s. In such situations, distinguishing between the disks likely becomes unnecessary, implying that the complexity of the above theorem drops to $O(n^3 K^2 \log^2 n) = O(n^3 \log^2 n)$. As described below, this is particularly evident when we are interested in finding locations which approximate the maximum lifetime.*

6.2 Approximating maximum lifetime

It must be noted that the obvious method of restricting the base-station location to that of one of the sensors itself can yield quite low lifetimes when compared with the optimum. For example, it is possible to guarantee a $(2+\varepsilon)^{-\alpha}$-approximation considering only the points in S itself as candidate base-station location. Indeed, assume that p, q, s is the triple defining $\mathbf{b}$ and τ^* in (7). Move $\mathbf{b}$ to the closest point from s. In the worst case, $\mathbf{b}$ is moved onto one of p, q, s, say, onto s; even in this case, the distance from $\mathbf{b}$ to any of p, q increases by at most a factor of 2 (and hence the lifetime goes down by at most 2^α). For each s from S, we can (approximately) maximize the lifetime by the algorithm from Proposition 4.7 or from Theorem 3.

The above factors which could get as bad as only guaranteeing only 6.25% when $\alpha = 4$ and only 25% efficiency when $\alpha = 2$ can be improved by using the properties proved in Section 4.1.1.

Recall that the ranges of only the followers may be increased to $(1 + \varepsilon)$ from 1 while incurring only a penalty factor of ε to the lifetime τ (Lemma 4.5). We thus consider the graph $G(1 + \varepsilon)$ where the bipartite graph may consist of edges upto length $(1 + \varepsilon)$.

Thus, from Lemma 4.6, we have only $O(\varepsilon^{-2})$ orbits around each sensor defining an arrangement of only $O(\frac{n^2}{\varepsilon^4})$ vertices. Applying the same technique as in Lemma 6.2, we get the following theorem:

Theorem 7. *The location for the base-station and the 2-level forwarding tree enabling transmission of data by all sensors for time* $(1-\varepsilon)\tau^*$ *where* τ^* *is the maximum system lifetime can be found in* $O(\frac{n^3}{\varepsilon^4}\log^2 n)$ *time.*

7 More General Network Topologies

So far, in the paper, we have dealt with the case when the transmissions from sensors have the following two restrictions: **(1)** each sensor is not allowed to split its data and transmit to multiple other sensors/base-station, i.e., the transmission scheme is a tree, and **(2)** the number of hops from any sensor to the base-station is at most 2. As mentioned earlier, such restricted transmission schemes provides two important benefits of low duty-cycling leading to better use of the energy and lower end-to-end delay in addition to a simpler clock synchronization process. In order to compare the performance of our algorithms with more general network transmission schemes, we now present two linear programs to deal with the finding optimal transmission schemes when these restrictions are not present. The algorithms in this section only deal with finding an optimal transmission scheme given a fixed base-station and *does not* deal with the problem of optimal base-station location when there are no or limited restrictions. It is interesting to note that Efrat *et al.* [13] presented $(1-\varepsilon)$-approximation algorithms for the problem of locating the base-station when no restrictions are present.

7.1 DAG: Lifting both restrictions

When *both* restrictions (1) and (2) are lifted, maximizing the lifetime can be formulated as a linear program (LP) [13, Lemma 1]. The LP works in the complete directed graph on $S \cup \mathbf{b}$. Each directed edge uv in the graph has an associated variable x_{uv} indicating the rate of the data flow on this edge (recall that when restriction (1) is lifted, the sensor is no longer required to relay all of its data via one other sensor; instead, any sensor can split its data between several other sensors). The other variable in the LP is $\mathcal{T} = 1/\tau$ where τ is the lifetime of the system (the LP objective is thus to minimize T). The complete LP formulation is given in Figure 6. Constraint (I) indicates that all data generated by the sensors must be transmitted to the base-station. Constraint (II) indicates that the out-flow from v equals to the in-flow plus the rate of data generation of v. Constraint (III) indicates that the total energy spent by v to transmit the data is at most 1, the capacity of v's battery (recall that in our model, receiving data is free). Clearly, x_{uv} cannot take negative values indicated by Constraint (IV).

Clearly, the transmission scheme produced by this LP is a *Directed Acyclic Graph* (DAG) and hence, we term this as the DAG algorithm.

7.2 Constrained-DAG: Putting restriction (2) back

We now solve the case when the restriction (1) is lifted, but the restriction (2) is active. That is, we want to maximize the system lifetime when the network is allowed to have arbitrary topology (not necessarily a tree), but the number of hops from every sensor to the base-station is at most 2.

$$\begin{aligned}
&\text{Minimize} && \mathcal{T} && && \\
&\text{subject to} && \sum_{u \in S} x_{u\mathbf{b}} = n && && \text{(I)} \\
& && x_{v\mathbf{b}} + \sum_{u \in S} x_{vu} = \sum_{u \in S} x_{uv} + 1 && \forall v \in S && \text{(II)} \\
& && \sum_{u \in S \cup \mathbf{b}} |vu|^{\alpha} x_{vu} \leq \mathcal{T} && \forall v \in S && \text{(III)} \\
& && x_{uv} \geq 0 && \forall u, v \in S && \text{(IV)}
\end{aligned}$$

Figure 6: LP Formulation for the DAG Algorithm

Thus, the transmission scheme is still a DAG but is now constrained to have paths of length at most 2 to the base-station. Hence, we call this algorithm as the Constrained-DAG algorithm.

To solve the problem we add to the above LP one constraint for each sensor $v \in S$:

$$\sum_{u \in S} x_{vu} \leq 1.$$

Together with the flow constraint (II) in Figure 6, the new constraint implies that

$$x_{v\mathbf{b}} \geq \sum_{u \in S} x_{uv}.$$

This forces v to send at least the same amount of data it receives directly to the base-station. However, it does not specify which data to send to the base-station. However, while doing the actual routing, it may simply send all incoming data directly while the rest is split according to the Constrained-DAG solution. Thus, any node v is free to choose where to send its *own* data – to another sensor or directly to the base-station; however, all data received from other sensors must be sent directly to $\mathbf{b}$. Thus, any feasible solution to the modified LP defines a (general) transmission scheme in which every sensor sends its data to the base-station in at most 2 hops.

8 Simulations

We experimentally compare transmission schemes which are two-trees and the transmission schemes produced by the two algorithms for general topologies presented in Section 7. In all models, it is assumed that the base-station location is given, and sensors continuously produce data over time. The three algorithms compared are:

- Two-Tree (TT): The data from every sensor must reach the base-station in at most 2 hops. The topology of the network must be a 2-Tree (tree with 2 levels), i.e., each sensor must send all data to one recipient. We use our algorithm from Section 4.2 to maximize system lifetime.
- Constrained-DAG (C-DAG): The data from every sensor must reach the base-station in at most 2 hops. The network can have arbitrary topology, i.e., any sensor can split its data between any number of other sensors. We use the modified LP from Section 7.2 to maximize system lifetime.

- DAG: There is no restriction on the number of hops in which data from a sensor must reach the base-station. The network can have arbitrary topology. We use the LP from [13, Section 2.1.2] (described here in Section 7.1) to maximize system lifetime.

Apart from the system lifetime, we measured two other output parameters:

Out-degree: The number of receivers to which a node transmits its data. For TT, the out-degree is always 1. A higher out-degree (meaning communication with more number of nodes) implies harder clock synchronization and higher duty cycling.

Hop count: The number of hops for a data packet from its origin sensor to the base-station. A higher hop count implies higher delay for the data to reach the base-station. For DAG and Constrained-DAG, we calculate the average hopcount of a unit of data (packet) to reach the base-station. For the DAG algorithm, this is caculated recursively as follows: The leaders (sensors transmitting all data directly to the base-station) have hop count 1. For a non-leader sensor v the hop count h_v is the average of the hop counts of its parents (nodes it forwards data to) weighted by the amount of transmitted data:

$$h_v = x_{vg} + \frac{\sum_{u \in S} h_u x_{vu}}{\sum_{u \in S} x_{vu}}$$

In case of the Constrained-DAG model, it may be calculated according to the following formula:

$$h_v = (x_{vg} - \sum_{u \in S \setminus g} x_{uv}) + 2 \sum_{u \in S \setminus g} x_{vu}$$

The first two terms measured the fraction of data transmitted directly and the last term measures the data which travels 2 hops. Here, we are making the assumption that once the Constrained-DAG solution is given, the sensors are able to distinguish the incoming packets from the self-generated data, thus all the incoming packets are directly sent to the base-station.

8.1 Setup

The sensor field is a square of side-length 10 units. The base-station is in the center of the square. The parameters of the experiments are the number of sensors n and the *Rayleigh fading factor* α. The sensors are placed in the field uniformly at random. We have conducted experiments for $n = 10, 20, \ldots, 150$ and for $\alpha = 2, 3, 4$. For each combination of the parameters, each experiment is repeated 20 times and the results are averaged.

The cost of sending a unit of data from a sensor u to a sensor v is $\max\{c_{\min}, |uv|^\alpha\}$ where the parameter $c_{\min}$ is introduced to reflect the fact that even if u and v are very close, a certain minimum energy is still required for the transmission. We set $c_{\min} = 1$.

8.2 Output

Sample outputs for all three models are presented in Figure 7. It can be see that in the TT model, our algorithm produces quite a sparse tree. For the Constrained-DAG output, the density of the edges is much higher. For the DAG, the density is about as small as for the TT, but the hop count is higher, with quite a few nodes at 5 or more hops from the base-station. Not surprisingly, under DAG and Constrained-DAG some nodes have high out-degree. Compared with DAG, the edges are more directed towards the base-station in the Constrained-DAG output.

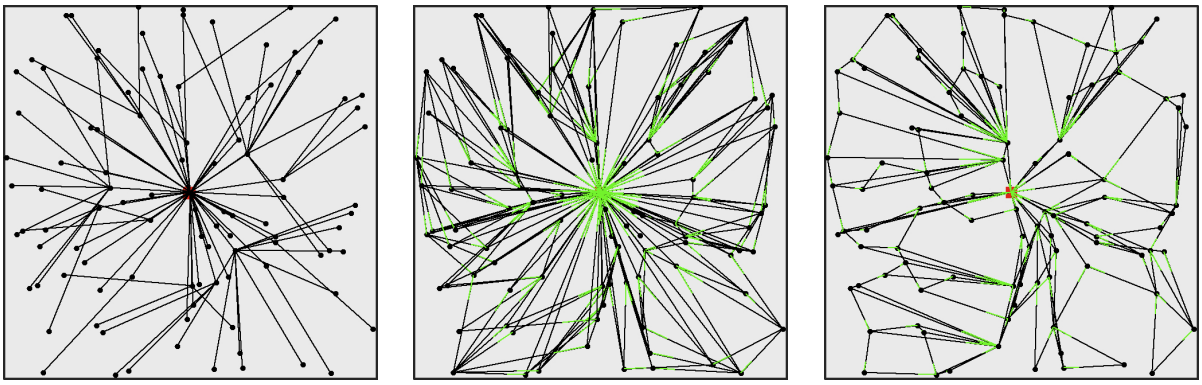


Figure 7: Output on the same instance for the TT (left), CONSTRAINED-DAG (middle) and DAG (right). An edge indicates non-zero data flow. For TT, all edges are directed towards the base-station; for CLP and LP, the green end of an edge indicates its direction.

To visualize the output of our algorithm for finding the optimal 2-tree, we implemented it in a Java applet available online at `http://www.cs.arizona.edu/people/taheri/?tab=research`. The applet allows the user to input sensor locations and graphically presents the tree.

8.3 Experimental results

The system lifetime as a function of the number of sensors is presented in Figure 8. It can be seen that the gap between the TT model and the other two models increases with increasing α. This can be attributed to the fact that as we increase α, the advantages of splitting data from a node are more apparent, making more complicated but also more power-efficient paths viable.

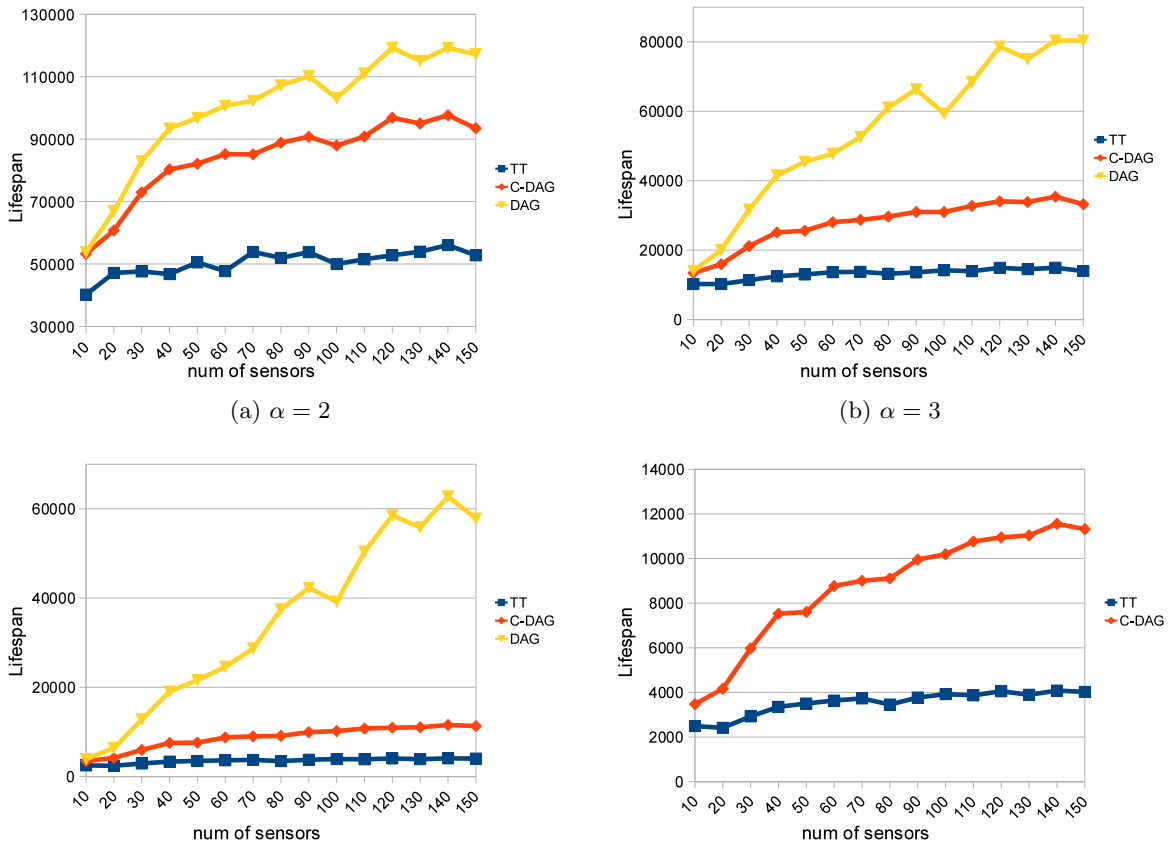
In the case of $\alpha = 2$, the graphs for CONSTRAINED-DAG and DAG seem to approach saturation. This leads us to infer that the lifetime under the CONSTRAINED-DAG and DAG is only a constant factor higher than for the TT model. It is interesting that this does not seem to hold for greater values of α. In particular, the bottom right plot in Figure 8 suggests that TT is perhaps not as effective as the CONSTRAINED-DAG in terms of the lifetime. Future research efforts could be directed towards exploring the effect of α on the lifetime.

		α		
		2	**3**	**4**
C-DAG/TT	***avg***	1.65	2.08	2.42
	max	1.84	2.38	2.83
DAG/TT	***avg***	1.94	4.03	9.23
	max	2.21	5.75	15.38

Table 2: The average and maximum ratios of lifetime under CONSTRAINED-DAG and DAG to that under TT.

Table 2 presents the ratio of the lifetime under CONSTRAINED-DAG and DAG to the lifetime under TT. The values are in line with the above observations.

Table 3 shows out-degree values for different models for $n = 100$ and $\alpha = 4$. As expected, both CONSTRAINED-DAG and DAG have higher out-degrees than TT. The higher the out-degree, the



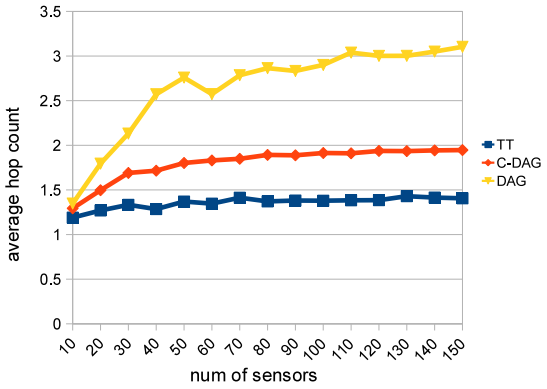
(a) $\alpha = 2$ (b) $\alpha = 3$

(c) $\alpha = 4$ (d) $\alpha = 4$, *only TT and C-DAG*

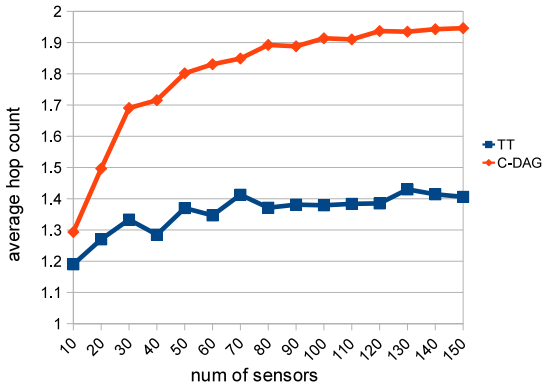
Figure 8: Lifetime as function of n. (d) shows the same graph as (c) but without the results for DAG in order to better compare TT and Constrained-DAG.

more complicated is the forwarding scheme, making synchronization between nodes more complicated. Hence, TT is seen to be best in this respect. It is interesting to note that Constrained-DAG has the highest out-degree of all the three. Since the number of hops is limited, in order to optimize the lifetime, Constrained-DAG has to split the data much more than DAG in order to avoid increasing the transmission rate of any single node too much. Thus, there is an inherent tradeoff in Constrained-DAG between the low delay and higher clock synchronization complexity together with higher duty cycling (since each node needs to communicate with more nodes than in Constrained-DAG).

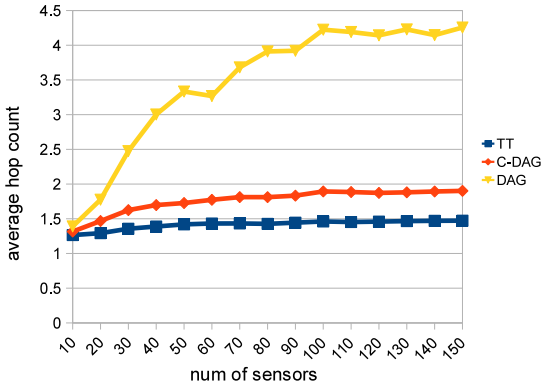
Figure 9 plots the hop count as function of n. Table 4 presents the average and maximum ratios of the hop-count of Constrained-DAG and DAG models over TT. As with lifetime, the gap in hop count between DAG and TT is higher for $\alpha = 3, 4$ than for $\alpha = 2$. TT has the lowest hop count, indicating that in terms of end-to-end delay, the TT model is much better than the DAG. Again, the gap between TT and Constrained-DAG is not very wide. Both these algorithms limit the hop count to 2; it appears that in the Constrained-DAG the actual hop-counts get closer to 2 with increasing n, whereas in the TT model the hop count does not go beyond 1.5 often. This indicates that, in practice, no more than half the nodes are designated as leaders in the TT



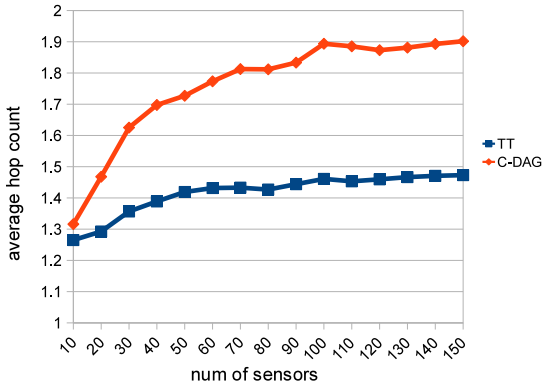
(a) $\alpha = 2$, *TT, C-DAG and DAG*



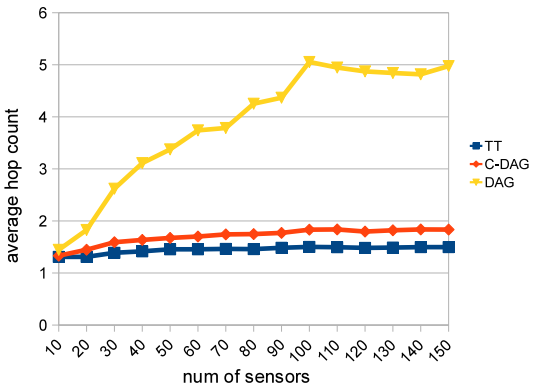
(d) $\alpha = 2$, *TT and C-DAG*



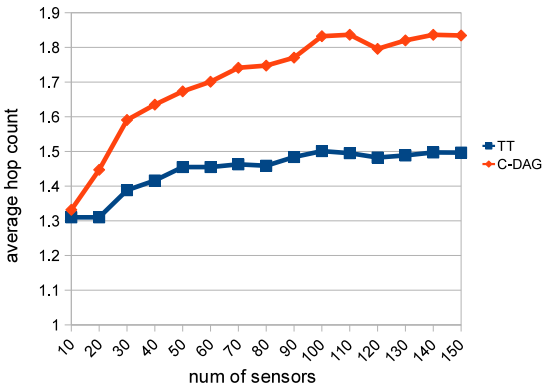
(b) $\alpha = 3$, *TT, C-DAG and DAG*



(e) $\alpha = 3$, *TT and C-DAG*



(c) $\alpha = 4$, *TT, C-DAG and DAG*



(f) $\alpha = 4$, *TT and C-DAG*

Figure 9: Hop count as a function of n for $\alpha = 2, 3, 4$.The right column ((d), (e) and (f)) shows the same plots as the left column((a), (b) and (c)) with DAG omitted, to better compare TT and Constrained-DAG.

	TT	C-DAG	DAG
average out-degree	1	2.33	1.91
max out-degree	1	15	8

Table 3: The average and maximum out-degrees.

algorithm.

		α		
		2	**3**	**4**
C-DAG/TT	***avg***	1.33	1.24	1.18
	max	1.40	1.30	1.23
DAG/TT	***avg***	1.94	2.42	2.64
	max	2.21	2.89	3.37

Table 4: The average and maximum ratio of hop-count under Constrained-DAG and DAG over that of TT.

In summary, the simulation results indicate that for the case of $\alpha = 2$, the deficiency in lifetime of TT may be offset by the improved hop count, i.e., better end-to-end delays. In particular, Constrained-DAG and DAG seem to be only a constant factor better than the TT in practice. However, for higher values of α, Constrained-DAG appears to be more effective than the TT and seems to be the best choice for higher values of α.

9 Conclusions

We presented algorithms to find transmission protocols for energy-constrained sensors. The main focus was on constructing the protocols under which the information from any sensor reaches the base-station after at most 2 hops. We gave efficient exact and approximation algorithms for two models: (1) when the transmission graph is restricted to be a tree, and (2) when the graph can be arbitrary. We compared experimentally the solutions found under the different models. Finally, we proved that when the transmission graph is allowed to be a tree with more than 2 layers, maximizing the system lifetime is NP-hard.

The running times of our exact solutions are quite high. When the basestation location is given, our approximation algorithm (Theorem 3) runs in nearly linear time ($O(\frac{n}{\varepsilon^2} \log n \log \frac{1}{\varepsilon})$), which is practical. On the other hand, for finding an (even approximately) optimal location for the basestation our solutions run in near-cubic time; we leave finding a faster algorithm open.

Another direction for future research is to consider our problems in the case when there is more than one base-station.

Acknowledgements

E. Arkin and J. Mitchell are partially funded by the National Science Foundation (CCF-07929019, CCF-1018388). A. Efrat is supported by NSF CAREER grant 0348000 and NSF grant CNS-1017714. V. Polishchuk is supported by the Academy of Finland, Grant 138520. We thank the anonymous reviewers for their many helpful comments and suggestions.

References

[1] M. Ajtai, J. Komlós, and E. Szemerédi. An 0(n log n) sorting network. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, STOC '83. ACM, 1983.

[2] K. Akkaya, M. Younis, and W. Youssef. Positioning of base stations in wireless sensor networks. *IEEE Communications Magazine*, 45(4):96–102, 2007.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422, 2002.

[4] T. Arampatzis, J. Lygeros, and S. Manesis. A survey of applications of wireless sensors and wireless sensor networks. In *Proceedings of the 13th Mediterranean Conference on Control and Automation*, MED '05. IEEE, 2005.

[5] E. M. Arkin, A. Efrat, J. S. B. Mitchell, V. Polishchuk, S. Ramasubramanian, S. Sankararaman, and J. Taheri. Data transmission and base-station placement for optimizing network lifetime. In *Proceedings of the 6th ACM SIGACT/SIGMOBILE International Workshop on Foundations of Mobile Computing*, DIALM-POMC '10. ACM, 2010.

[6] A. Bogdanov, E. Maneva, and S. Riesenfeld. Power-aware base station positioning for sensor networks. In *Proceedings of 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, INFOCOM '04. IEEE, 2004.

[7] C. Buragohain, D. Agrawal, and S. Suri. Power aware routing for sensor databases. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, INFOCOM '05. IEEE, 2005.

[8] J.-H. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(4):609–619, 2004.

[9] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *Journal of the ACM*, 34(1):200–208, 1987.

[10] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to algorithms*. The MIT press, 2001.

[11] Y. Diao, D. Ganesan, G. Mathur, and P. Shenoy. Rethinking data management for storage-centric sensor networks. In *Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research*, CIDR '07, 2007.

[12] Y. Dinitz. Dinitz' algorithm: The original version and even's version. In O. Goldreich, A. Rosenberg, and A. Selman, editors, *Theoretical Computer Science*, volume 3895 of *Lecture Notes in Computer Science*, pages 218–240. Springer Berlin / Heidelberg, 2006.

[13] A. Efrat, S. Har-Peled, and J. Mitchell. Approximation algorithms for two optimal location problems in sensor networks. In *Proceedings of the 2nd International Conference on Broadband Networks*, BROADNETS '05. IEEE, 2005.

[14] A. Efrat, A. Itai, and M. J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, 2001.

[15] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *Proceedings of the 15th International Parallel & Distributed Processing Symposium*, IPDPS '01. IEEE, 2001.

[16] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, 36:147–163, 2002.

[17] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[18] Y. Gu, D. Bozdağ, R. W. Brewer, and E. Ekici. Data harvesting with mobile elements in wireless sensor networks. *Computer Networks*, 50(17):3449–3465, 2006.

[19] W. B. Heinzelman. *Application-specific protocol architectures for wireless networks*. PhD thesis, Massachusetts Institute of Technology, 2000.

[20] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[21] D. Jea, A. Somasundara, and M. Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. In *Proceedings of the 1st IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS '05. IEEE, 2005.

[22] M. Kodialam and T. Nandagopal. Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, MobiCom '03, pages 42–54, New York, NY, USA, 2003. ACM.

[23] V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. Algorithmic aspects of capacity in wireless networks. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '05, pages 133–144, New York, NY, USA, 2005. ACM.

[24] Q. Li and D. Rus. Global clock synchronization in sensor networks. *IEEE Transactions on Computers*, 55(2):214–226, 2006.

[25] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. In *Proceedings of the 5th IEEE International Symposium on Information Processing in Sensor Networks*, IPSN '06. IEEE, 2006.

[26] N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.

[27] J. Pan, L. Cai, Y. Hou, Y. Shi, and S. Shen. Optimal base-station locations in two-tiered wireless sensor networks. *IEEE Transactions on Mobile Computing*, 4(5):458–473, 2005.

[28] T. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[29] K. Römer. Time synchronization in ad hoc networks. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, MobiHoc '01. ACM, 2001.

[30] Y. Shi and Y. Hou. Theoretical results on base station movement problem for sensor network. In *Proceedings of the 27th Annual Joint Conference of the IEEE Computer and Communications Societies*, INFOCOM '08. IEEE, 2008.

[31] Y. Shi, Y. Hou, and A. Efrat. Algorithm design for base station placement problems in sensor networks. In *Proceedings of the 3rd international conference on Quality of service in heterogeneous wired/wireless networks*, QSHINE '06. ACM, 2006.

[32] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *IEEE Network*, 18(4):45 – 50, 2004.

[33] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava. Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In *Proceedings of the 25th IEEE International Real-Time Systems Symposium*, RTSS '04. IEEE, 2004.

[34] W. Su and I. F. Akyildiz. Time-diffusion synchronization protocol for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 13(2):384–397, 2005.

[35] B. Sundararaman, U. Buy, and A. D. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3(3):281 – 323, 2005.

[36] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292 – 2330, 2008.