Building Maximum Lifetime Shortest Path Data Aggregation Trees in Wireless Sensor Networks

MENGFAN SHAN, GUIHAI CHEN, DIJUN LUO, XIAOJUN ZHU, and XIAOBING WU,
Nanjing University

In wireless sensor networks, the spanning tree is usually used as a routing structure to collect data. In some situations, nodes do in-network aggregation to reduce transmissions, save energy, and maximize network lifetime. Because of the restricted energy of sensor nodes, how to build an aggregation tree of maximum lifetime is an important issue. It has been proved to be NP-complete in previous works. As shortest path spanning trees intuitively have short delay, it is imperative to find an energy-efficient shortest path tree for time-critical applications. In this article, we first study the problem of building maximum lifetime shortest path aggregation trees in wireless sensor networks. We show that when restricted to shortest path trees, building maximum lifetime aggregation trees can be solved in polynomial time. We present a centralized algorithm and design a distributed protocol for building such trees. Simulation results show that our approaches greatly improve the lifetime of the network and are very effective compared to other solutions. We extend our discussion to networks without aggregation and present interesting results.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Wireless sensor networks, media access control, maximum lifetime, load balance

ACM Reference Format:
Mengfan Shan, Guihai Chen, Dijun Luo, Xiaojun Zhu, and Xiaobing Wu. 2014. Building maximum lifetime shortest path data aggregation trees in wireless sensor networks. ACM Trans. Sensor Netw. 11, 1, Article 11 (June 2014), 24 pages.
DOI: http://dx.doi.org/10.1145/2629662

1. INTRODUCTION

Data collection is an important operation in many wireless sensor network (WSN) applications, such as structure maintenance [Xu et al. 2004], environment monitoring [Liu et al. 2013], and habitat monitoring [Mainwaring et al. 2002]. Some divisible functions [Giridhar and Kumar 2005] (e.g., SUM, MAX, MIN, AVERAGE) are widely

---

A preliminary version of this article was presented in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'11)* [Luo et al. 2011].
This work was partly supported by the National Natural Science Foundation of China under grants 61133006, 61321491, and 61373130 and the National Basic Research Program of China (973 Program) under grants 2014CB340300 and 2012CB316200. X. Zhu was partly supported by the program B for Outstanding Ph.D. Candidate of Nanjing University under grant 201301B014.
Authors' addresses: M. Shan, G. Chen (corresponding author), D. Luo, X. Zhu, and X. Wu, State Key Lab for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China, X. Zhu is also with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China; email: nju.mfshan@gmail.com, gchen@nju.edu.cn, {luodijun, gxjzhu}@gmail.com, and wuxb@nju.edu.cn.

DOI: http://dx.doi.org/10.1145/2629662

used to fuse data and avoid energy holes [Wu et al. 2008a]. Different from clustering schemes [Ye et al. 2005; Chen et al. 2009], many previous works use a tree structure (e.g., Gnawali et al. [2009], and Liang et al. [2009]) to collect data. A sensor node receives messages from its children and forwards the received messages and its own sensing message to its parent. Since wireless sensor nodes are battery powered and usually deployed in harsh environments, how to construct an efficient tree to prolong the lifetime of the network is an important problem.

Wu et al. [2008b] prove that finding a data aggregation tree for data collection with maximum lifetime in WSNs is NP-complete. They propose an approximation algorithm that produces a suboptimal tree. Although this solution is close to optimal, it may result in long delivery delay if the tree is deep. Consequently, it might not be suitable for time-critical applications. A shortest path tree is a tree in which the hops to the sink are minimum for each node. Many previous works [Chen et al. 2005; Huang et al. 2007; Wan et al. 2009] that focus on minimizing the latency in data aggregation networks are based on shortest path trees. Intuitively, a shorter path usually brings shorter delay. Meanwhile, as detailed in Section 7, we find that in WSNs *without* in-network aggregation, the problem of building a shortest path tree with the maximum lifetime is NP-complete as well.

We study the problem of building a shortest path data aggregation tree with maximum lifetime for WSNs. We find one shortest path aggregation tree for each sensor network that would maximize network lifetime. We find that this problem can be reduced to a general version of *semi-matching* problems [Fakcharoenphol et al. 2010; Harvey et al. 2003] and show that it can be solved in polynomial time. Our main contributions are as follows (a preliminary version of this work appeared in Luo et al. [2011]):

—To the best of our knowledge, we are the first to study the problem of maximizing lifetime for the shortest path aggregation tree. The restriction to shortest path trees comes from the requirement of delay. We find that when it is restricted to shortest path trees, the original NP-complete problem is in P.
—We give a centralized algorithm that runs in $O(|E|\sqrt{N}\log N)$ time. This algorithm, to the best of our knowledge, is the fastest in the literature. This algorithm is designed according to our finding that the problem can be divided into several subproblems, each of which is a general version of a semi-matching problem.
—For better applicability of our result to WSNs, we present a distributed protocol. Since it is hard to transform the fastest centralized algorithm to a distributed one, we use another algorithm (different from the aforementioned centralized algorithm) for the semi-matching problem that has slightly high time complexity but is more suitable for distributed implementation.
—We carry out extensive simulations to evaluate our approaches. The results show that our approaches greatly improve the lifetime of the network.

The rest of this article is organized as follows. Section 2 reviews some related works. Section 3 introduces the model and formulates the problem. We present the centralized algorithm in Section 4 and a distributed protocol in Section 5. Section 6 gives the simulation results. In Section 7, we discuss the same problem when there is no aggregation in networks. Finally, we conclude our work in Section 8.

2. RELATED WORKS

There are many related works on the problem of building a shortest path data aggregation tree with maximum lifetime in WSNs. In this section, we classify the related works into four categories—namely, non–tree-based schemes, general tree-based schemes,

shortest path tree-based schemes, and dynamic routing protocols. The first three categories of schemes are also referred to as static schemes in that these schemes are proposed to optimize the network lifetime given a specific network topology. Hence, the network lifetime in such schemes is usually defined as the duration from the beginning of network operation until the first node runs out of energy. We refer to the dynamic routing protocols as schemes proposed to adjust their routing structures dynamically according to the topology change. The network topology changes when some nodes use up their energy. Therefore, in such protocols, network lifetime is usually defined as the time span from the start of network operation until a certain percent of nodes deplete their energy.

Non–tree-based schemes for maximizing network lifetime in data collection are proposed in Park and Sahni [2006] and Shi and Hou [2008]. Park and Sahni [2006] prove that maximizing lifetime routing in WSN is NP-hard and propose a heuristic approach. They do not consider the energy consumption for receiving messages. Shi and Hou [2008] take advantage of mobile sink for data collection to prolong the network lifetime.

Tree routing structures appear in many data gathering and aggregation protocols. Wu et al. [2010] focus on the energy-efficient wake-up schedule in the data aggregation process. They also propose effective algorithms to construct data aggregation trees such that both the energy consumption and the network throughput are within a constant factor of the optimal. The total energy consumption of all sensor nodes is also an important issue in data aggregation networks. Kuo and Tsai [2012] analyze this problem under some certain aggregation ratio q. They prove that this problem is NP-complete and that every shortest path tree has an approximation ratio of 2. When there are pure relay nodes in the network, the problem is proved to be NP-complete and a seven-approximation algorithm is proposed. Recently, there have been some works about constructing a tree routing structure with maximum lifetime. Tan and Körpeoğlu [2003] propose two algorithms based on sensor locations and minimum spanning tree construction, and one is the power-aware version of the other. Xue et al. [2005] use a linear programming approach to give an approximation algorithm for finding a maximum lifetime aggregation tree. Wu et al. [2008b] prove that constructing an arbitrary aggregation tree with the maximum lifetime is NP-hard and propose an approximation algorithm. In contrast, our scheme focuses on how to construct a shortest path aggregation tree with maximum lifetime.

There are many works using a shortest path tree to collect data from the network [Le et al. 2007; Huang et al. 2007; Wan et al. 2009; Liang et al. 2009]. Le et al. [2007] propose an approach to construct a shortest path tree for every sink and dynamically distribute dataflow among different paths to the sinks to balance the load among the sinks. Huang et al. [2007] and Wan et al. [2009] investigate the latency issue of data aggregation trees. They present centralized and distributed approaches for minimum latency data scheduling in the data aggregation tree, which is proved to be NP-complete in Chen et al. [2005]. Liang et al. [2009] define the length of edges as the expected transmission count (ETX) and construct a shortest path tree for each channel. Similar to these schemes, shortest path tree is used in our work. But we assume that there is one single sink in the networks and do not consider load balancing among multiple sinks. We consider the problem of how to build an optimal shortest path data aggregation tree, and the delivery latency is not our focus.

Other schemes for data collection are dynamic routing protocols [Gnawali et al. 2009; Baydere et al. 2006]. Collection tree protocol (CTP) is proposed in Gnawali et al. [2009]. It outputs anycast routes to the sink. Baydere et al. [2006] give a dynamic path switching algorithm for event-driven data collection. They aim to improve the reliability of the network and give lifetime analysis. Our proposed work belongs to the category

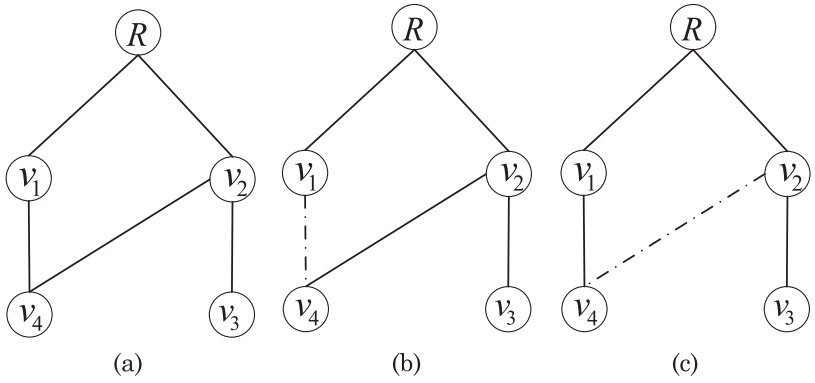

Fig. 1. A simple example for the optimal shortest path tree.

of static scheme, as its aim is to optimize the network lifetime given a specific topology. For static schemes, the lifetime is usually defined as the duration from the beginning of network running until the first node dies. However, most dynamic data collection protocols do not care when the first node dies. They adjust the routing structure when the network topology changes. Theoretical optimality is the main contribution of this work. In this article, we present an algorithm for finding the optimal shortest path data aggregation tree with maximum network lifetime in WSNs. The output of our algorithm is a data aggregation tree based on the given network topology and initial energy of nodes. The problem of how to build a data aggregation tree with maximum lifetime in WSNs has been proved to be NP-complete [Wu et al. 2008b]. We find that this problem is in fact polynomially solvable when the data aggregation tree is restricted to a shortest path tree. Given specific network topology and initial energy of nodes, our scheme gives the optimal tree in terms of lifetime. As shown in simulation results later, the proposed scheme has comparable or better performance in terms of network lifetime when compared to dynamic data collection protocols, although theoretical optimality is the main focus of this work and the proposed scheme is designed under different definition of network lifetime.

3. NETWORK MODEL AND PROBLEM FORMULATION

In this article, we only consider the energy depletion in transmitting and receiving messages, which are the two major energy consuming operations [Wu et al. 2008b]. In this section, we first present the model and the assumption of the network, then formulate the problem as a load balancing problem.

3.1. A Motivating Example

In this subsection, we illustrate our problem by a simple example.

Figure 1(a) shows a small WSN. There are five nodes and five edges, and each node can do in-network aggregation. There are two possible shortest path trees, as shown in Figure 1(a) and (c) (each solid line represents a tree edge). Assume that the root node R has infinite energy and that nonroot nodes v_1, v_2, v_3, v_4 have initial energy 2, 7, 3, 3, respectively. Further assume that each transmission and reception consumes 1 unit of energy.

We define the lifetime of a tree as the number of data collection rounds that it can support. In a data collection round, a sensor node collects packets from its children, aggregates with its own sensing information, and sends exactly one packet to its parent. We assume a reliable network transmission in each transmission. Following this definition, the tree in Figure 1(b) has a lifetime of 2. To see this, note that in each round, v_2 receives two messages from v_3 and v_4, aggregates them with its local data,

Table I. Terminology

$G(V, E)$	Graph of the network: V represents the set of nodes, whereas E corresponds to the set of edges
T	Aggregation routing tree that is a shortest path tree and the sink is the root
N	Number of nodes in the network, $N = \|V\|$
Tx	Energy consumption for transmitting a message
Rx	Energy consumption for receiving a message
$E(i)$	Nonreplenishable energy for node i, $i = 1, 2, \ldots, N$, $E(0) = \infty$ for the sink
$C(T, i)$	Number of children for the node i in tree T
$l(T, i)$	Lifetime of the node i in a routing tree T
$l(T)$	Lifetime of a routing tree T, i.e., $l(T) = \min_{i=1,\ldots,N} l(T, i)$
$L(T, i)$	Load of the node i in a routing tree T, defined as $L(T, i) = \frac{1}{l(T,i)}$
$L(T)$	Load of the network under a routing tree T, i.e., $L(T) = \max_{i=1,\ldots,N} L(T, i)$
$h(i)$	Minimum number of hops from the node i to the sink, $h(0) = 0$
d	Maximum value of $h(i)$, which is the height of the fat tree
L_h	Minimum of the maximum load for the nodes at height h
SPT	Set of the shortest path trees in the network
N_h	Set of nodes at height h
E_h	Set of edges between nodes at height h and $h + 1$

and then transmits the new data message to the sink, so it consumes 3 units of energy; the other nodes only consume 1 unit of energy in one round. After two rounds, node v_1 depletes its energy and dies, resulting in the lifetime of 2. For the tree in Figure 1(c), v_4 is the child of v_1 instead of v_2. In each round, the energy consumption of v_1, v_2, v_3, v_4 is 2, 2, 1, 1, respectively, so the network's lifetime is 1. As a result, we prefer the shortest path tree in Figure 1(b) to the one in Figure 1(c).

3.2. Network Model

There are N sensor nodes $1, 2, \ldots, N$ and a sink 0 in the network. We consider the network as an undirected graph $G(V, E)$, in which $V = \{0, 1, 2, \ldots, N\}$ represents the set of N sensor nodes and the sink (labeled as 0), and E is the set of edges in the network. There is an edge between two nodes if and only if they can communicate with each other directly. Each sensor node $i \in V$ has different and nonreplenishable energy $E(i)$ ($E(i) \geq 0$), whereas the sink has infinite energy $E(0) = +\infty$. Nodes consume the same energy Tx to transmit a message and Rx to receive a message. We assume that the network has applied multifrequency technology and TDMA-like MAC to avoid overhearing and collision so that the energy consumption contains only two parts: transmitting messages and receiving messages.

The network is connected so that each node has at least one path to the sink. The routing structure is a static tree to be found. Given a tree, nodes conduct in-network aggregation: at any round, each node aggregates the received messages from its children, combines them with its own message into a single outgoing message, and then sends the combined message to its parent.

We list the notations used in this article in Table I. In each round, a node generates one message and transmits it to the sink via a single-hop or multihop path. So the total energy consumption for a node in a round is the sum of energy depletion for transmitting exactly a message to its parent and that for receiving a message from each of its children. If a node i has $C(T, i)$ children under a special routing tree T, then in each round the node consumes energy $T_X + R_X \cdot C(T, i)$. It follows that the lifetime l of the node i in the routing tree T can be represented as the total number of rounds

that the node can support. We have

$$l(T,i) = \frac{E(i)}{Tx + Rx \cdot C(T,i)}. \tag{1}$$

We define the lifetime of the network as the time until the first node depletes its energy. This definition is widely used in the literature (e.g., Wu et al. [2008b]). We assume that only one aggregation tree is given for each sensor network and that no dynamic change is allowed in this article. We have the lifetime of the network under a routing tree T

$$l(T) = \min_{i \in V} l(T,i). \tag{2}$$

3.3. Problem Formulation

Based on our model, we can now formulate the problem:

PROBLEM 1 (MLST). *Given a graph $G(V,E)$ and each nonroot node i's initial energy $E(i)$ for $i = 1, 2, \ldots, N$, find a maximum lifetime shortest path tree (MLST) T_{opt} from the set of shortest path trees in the graph.*

Formally, we have

$$l(T_{opt}) = \max_{T \in SPT} l(T) = \max_{T \in SPT} \min_{i \in V} l(T,i).$$

However, it is hard to directly solve this problem. We transform it into another formulation as follows. Define the *load* of a node as the inverse of its lifetime:

$$L(T,i) = \frac{1}{l(T,i)}.$$

Denote by $L(T)$ the load of the tree T, and define $L(T) = \max_i L(T,i)$, then $L(T) = \max_i L(T,i) = \max_i \frac{1}{l(T,i)} = \frac{1}{\min_i l(T,i)} = \frac{1}{l(T)}$.

PROBLEM 2 (MINST). *Given a graph $G(V,E)$ and each non-root node i's initial energy $E(i)$ for $i = 1, 2, \ldots, N$, find a minimum load shortest path tree (MinST) T'_{opt} among all the shortest path trees. Formally,*

$$L(T'_{opt}) = \min_{T \in SPT} L(T) = \min_{T \in SPT} \max_{i \in V} L(T,i).$$

THEOREM 1. *MLST problem is equivalent to MinST.*

PROOF. Since $l(T'_{opt}) = \frac{1}{L(T'_{opt})} = \frac{1}{\min_T L(T)}$ and $\frac{1}{\min_T L(T)} = \max_T \frac{1}{L(T)} = \max_T l(T)$, T'_{opt} is a tree that has the maximum lifetime. Similarly, we can prove that T_{opt} is a tree that has the minimum load. □

The advantage of MinST formulation is that we can decompose it into simpler subproblems, as shown in the next section. In the rest of the article, we will focus on this MinST problem.

4. A CENTRALIZED APPROACH

To solve the MinST problem, we first construct a fat tree [Wu et al. 2008c] from the network and show that every edge of a shortest path tree belongs to the fat tree. Then, we divide the MinST problem into subproblems, each of which is a general version of semi-matching [Harvey et al. 2003]. We will give the centralized algorithm next. It

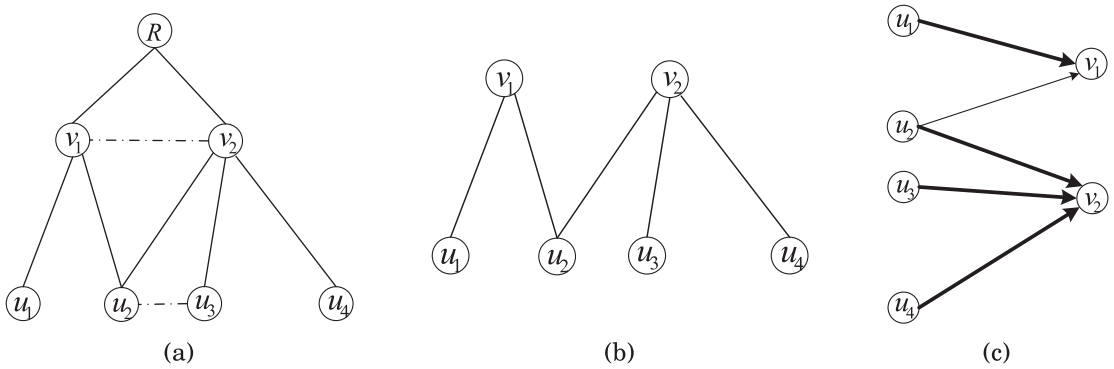

Fig. 2. (a) The fat tree of the graph. Edge(v_1, v_2) does not appear in any shortest path tree. (b) The nodes with same height in the fat tree; v_1 and v_2 have the same height. Their candidate children are also at the same height. (c) The general version of semi-matching. Each vertex u is incident with exactly one edge in semi-matching.

solves the load balance semi-matching problem by the min-cost max-flow approach and runs in $O(|E|\sqrt{N}\log N)$ time, which is the fastest to the best of our knowledge.

4.1. Fat Tree

We apply a breadth-first search algorithm [Leiserson et al. 2001] to construct a fat tree rooted at the sink. Each node stores its height (hop counts to the sink), as well as the nodes of the next hops, along the shortest paths to the sink. Therefore, the fat tree is the union of all shortest path trees, where branches from the sink to each node are paths with the least hop count. Any edge between two nodes at the same height will be removed from the graph, as it will never appear in any shortest path tree.

Consider the example in Figure 2(a). The fat tree consists of solid edges. The two dashed edges (v_1, v_2) and (u_2, u_3) are removed. Note that the edge (v_1, v_2) does not appear in any shortest path tree since the height of node v_1 is equal to that of node v_2. Similarly, the edge (u_2, u_3) does not appear in any shortest path tree. So by using fat tree, we reduce the number of edges, simplifying the graph.

Therefore, it is safe to only consider the fat tree, instead of the original graph. Then, finding a shortest path tree is equivalent to assigning a parent for each node from the fat tree, or assigning a set of neighbors in the fat tree as children for each node.

4.2. Children Assignment and Load Balancing

Our goal is to find a MinST T'_{opt} subject to $L(T'_{opt}) = \min_T \max_{i \in V} L(T, i)$. Recall that

$$L(T, i) = \frac{1}{l(T, i)} = \frac{Rx \cdot C(T, i) + Tx}{E(i)}. \tag{3}$$

Since Tx and Rx are fixed for each node, in fact, $C(T, i)$ and $E(i)$ determine the value of $L(T, i)$. Therefore, given a node, its load is only affected by its number of children. The number of children of a node, on the other hand, is only influenced by the children assignment of nodes at the same height. Thus, we have the following lemma.

LEMMA 1. *For a node with height h, changing its load only affects loads for the nodes at the same height.*

PROOF. Note that the conflict in children assignment only happens among the nodes with the same height, since their candidate children are at the same height. This implies that the change of the load for node i only affects the loads of the nodes at the same height with i, which is $N_h - \{i\}$. □

For example, in Figure 2(b), the height of top row nodes (i.e., v_1 and v_2) is 1 in the fat tree (Figure 2(a)) and node v_1 has a set of candidate children: $\{u_1, u_2\}$. Node v_2 has a set of candidate children: $\{u_2, u_3, u_4\}$. So u_2 has two candidate parents: v_1 and v_2. Node v_1 competes with node v_2 to choose u_2 as a child.

Lemma 1 shows that we can do children assignment independently at each height. Specifically, we can solve the MinST problem in parallel at each height: for the nodes in N_h, we try to find a children assignment t_h such that the maximum load is minimized (denote L_h as the minimum value of the maximum load at height h). We combine these children assignments $T = \cup_{h=0,1,\ldots,d} t_h$. Consequently, T is a shortest path tree, and the maximum load of the nodes in N_h is L_h. We have $L_h \leq L(T_{opt})$; otherwise, the assignment from T_{opt} will give an even lower load, contradicting the definition of L_h. So we have

$$L(T) = \max_{h=0,1,\ldots,d} L_h \leq \max_{h=0,1,\ldots,d} L(T_{opt}) = L(T_{opt}).$$

Additionally, T_{opt} has the minimum load, which implies that $L(T_{opt}) \leq L(T)$. So $L(T) = L(T_{opt})$.

From preceding analysis, the MinST problem can be divided into subproblems.

PROBLEM 3 (SUBPROB). *Given a bipartite graph $G_h = (N_{h+1} \cup N_h, E_h)$ as in Figure 2(c), where N_{h+1} is the set of left-hand vertices, N_h is the set of right-hand vertices, and $E_h \subseteq N_{h+1} \times N_h$. Each node $v \in N_h$ has energy $E(v)$. We define a set $M \subseteq E_h$ as a semi-matching if each vertex $u \in N_{h+1}$ is incident with exactly one edge in M. For any $v \in N_h$, let $deg_M(v)$ be the number of incident edges in M, then the load of v in M is defined as*

$$L(M, v) = \frac{Tx}{E(v)} + \frac{Rx}{E(v)} \cdot deg_M(v). \tag{4}$$

The load of M is $L(M) = \max_{v \in N_h} L(M, v)$. The problem is to find an semi-matching for G_h with minimum load.

Therefore, we only need to solve the SubProb and after combining the solutions of all SubProb, we get the solution of the MinST problem. Note that our definition is different from the definition of semi-matching in Harvey et al. [2003], where the load of each node v is $deg_M(v)$. Observe that if $Tx = 0$ and $E(v) = Rx$ for $v \in N_h$, the load in SubProb is simplified to $deg_M(v)$. So our SubProb is a generalized version of the semi-matching problem.

4.3. A Min-Cost Max-Flow Approach

In this subsection, we present the centralized algorithm, which is to the best of our knowledge the fastest in the literature. This algorithm is extended from the fastest algorithm in Fakcharoenphol et al. [2010] for solving the traditional semi-matching problem. The basic idea is to construct an equivalent min-cost max-flow problem for a given SubProb.

Similar to Fakcharoenphol et al. [2010], we show how to construct a network G_N such that a min-cost max-flow determines an optimal semi-matching in SubProb.

We construct a network G_N from $G_h = (N_{h+1} \cup N_h, E_h)$ by adding $2|N_h| + |E_h| + 2$ nodes and $4|N_h| + |N_{h+1}| + 2|E_h|$ edges (Figure 3). The additional vertices are a source S, a sink T, a set of vertices $P = \{p_1, \ldots, p_{|N_h|}\}$, and a set of "cost centers" $C = \{c_1, c_2, \ldots, c_{|E_h|}, \ldots, c_{|E_h|+|N_h|}\}$. Edges with cost 0 and capacity 1 connect S to each of the vertices in N_{h+1} and P. And edges in the original graph G_h between N_{h+1} and N_h are directed from N_{h+1} to N_h with cost 0 and capacity 1. Each vertex p in P is connected to a unique vertex v in N_h with cost 0 and capacity 1. Each vertex v in N_h is connected

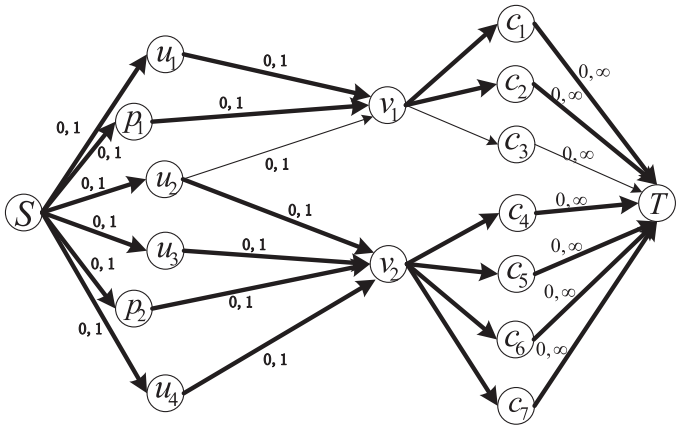

Fig. 3. The corresponding min-cost max-flow problem. Some edges are labeled with an entry (cost, capacity). Bold edges have flow of unit 1.

to $deg(v) + 1$ distinct cost centers $c_0^v, c_1^v, c_2^v, \ldots, c_{deg(v)}^v$ with edges of capacity 1 and costs $\frac{Tx}{E(v)}, \frac{Tx+Rx\times 1}{E(v)}, \ldots, \frac{Tx+Rx\times deg(v)}{E(v)}$. So there are $|E_h| + |N_h|$ cost centers in total. For each cost center c in C, it is connected to T with edges of cost 0 and capacity infinite.

We denote the residual graph with respect to a flow f by R_f. If a path p between two cost centers c_i and c_j in R_f exists and is a negative path, then we call p a *cost-reducing path*.

LEMMA 2. *If f is integral and is a maximum flow in G_N, then f determines a semi-matching in G_h.*

PROOF. Observe that all edges from S to $N_{h+1} \cup P$ have unit capacity, so $\{(S, u)\big|u \in N_{h+1} \cup P\}$ is a cut with capacity $|N_{h+1}| + |N_h|$. We first show that there exists a flow that makes this cut saturated. Let's consider the following flow. Edges of the cut have the flow of 1, and each node $u \in N_{h+1} \cup P$ sends one unit of flow to an adjacent node $v \in N_h$. Node v sends the inflow via its corresponding cost centers to the sink. It is easy to verify that this flow makes the cut $\{(S, u)|u \in N_{h+1} \cup P\}$ saturated. By the max-flow min-cut theorem [Leiserson et al. 2001], the value of the maximum flow is $|N_{h+1}|+|N_h|$. So if f is integral and is a maximum flow in G_N, then the cut is saturated. Since each node $u \in N_{h+1} \cup P$ has the inflow of 1 from source S, it must send one unit flow to a single node $v \in N_h$. Therefore, each node $u \in N_{h+1}$ is matched with a node $v \in N_h$, and f determines a semi-matching in G_h. □

We borrow the ideas from Harvey et al. [2003] and prove the following theorem.

THEOREM 2. *If a max-flow f is a min-cost flow in G_N, then the corresponding semi-matching A in graph G_h is optimal, which has the minimum value of the maximum load.*

PROOF. First we present the following claims as in Fakcharoenphol et al. [2010].

CLAIM 1. *A max-flow f is a min-cost flow in G_N if and only if there is no cost-reducing path in $R_f(G_N)$.*

Details of the proof can be found in Fakcharoenphol et al. [2010], so we omit it here.

By Claim 1, R_f has no cost-reducing path. We show that a cost-reducing path must exist in R_f if A is not optimal.

Let O be an optimal semi-matching that minimizes the maximum load of G_h, chosen that the symmetric difference $O \bigoplus A = (O \backslash A) \cup (A \backslash O)$ is minimized. Let

$L(A) = \max_{v \in N_h} L(A, v)$ and $L(A, v_0) = L(A)$. Assume that A is not optimal, which implies that A has greater maximum load than O: $L(O) < L(A)$. We use $deg_O(v)$ and $deg_A(v)$ to denote the number of N_{h+1} vertices assigned to v by O and A, respectively. Let G_d be the subgraph of G_h induced by the edges of $O \bigoplus A$. Color the edges of $O \backslash A$ with green and the edges of $A \backslash O$ with red. Direct the green edges from N_{h+1} to N_h and the red edges from N_h to N_{h+1}.

CLAIM 2. *The graph G_d is acyclic.*

PROOF. Let $P = (v_1, u_1 \ldots, v_2, u_2, v_1)$ be a cycle in G_d. By alternating the match of the N_{h+1}-vertices along P, we get an optimal semi-matching with smaller symmetric difference from A, which contradicts the choice of O. Thus, the graph G_d is acyclic. □

Since $L(A, v_0) = L(A)$ and $L(A) > L(O)$, we must have $L(A, v_0) > L(O, v_0)$. Starting from v_0, we construct an alternating red-green path P' as follows:

(1) From an arbitrary vertex $v \in N_h$, if $deg_{A \backslash O}(v) > 0$ and $L(A, v) + \frac{Rx}{E(v)} \geq L(A, v_0)$, we construct P' by following an arbitrary red edge directed out from v.
(2) From an arbitrary vertex $u \in N_{h+1}$, we construct P' by following the single green edge directed out from u, which must exist.
(3) For other cases, we stop.

Since G_d is acyclic, P' must be well defined and finite. Let $v_2 \in N_h$ be the final vertex on the path. Then, there must be two cases

(a) $L(A, v_2) + \frac{Rx}{E(v_2)} < L(A, v_0)$. Since f is a min-cost max-flow in the network, the edges $(v_2, c_0^{v_2})$,$(v_2, c_1^{v_2})$, …, $(v_2, c_{deg_A(v_2)}^{v_2})$ are saturated. For edges $(v_2, c_k^{v_2})$ where $deg_A(v_2) < k \leq deg(v_2)$, they have flow 0. So in graph R_f, there is a cost center c_2 adjacent to v_2 such that the cost of edge (v_2, c_2) is $L(A, v_2) + \frac{Rx}{E(v_2)}$; there is a cost center c_1 adjacent to v_0 such that the cost of edge (c_1, v_0) is $-L(A, v_0)$. The path (c_1, v_0, P', v_2, c_2) is a cost-reducing path in R_f, which contradicts the definition of f.
(b) $deg_{A \backslash O}(v_2) = 0$, then we have $deg_A(v_2) \leq deg_O(v_2)$. Since P' arrives at v_2 via a green edge, we have $deg_{O \backslash A}(v_2) \geq 1$, so $1 + deg_A(v_2) \leq deg_O(v_2)$. Thus, we have

$$L(A, v_2) + \frac{Rx}{E(v_2)} \leq L(O, v_2) \leq L(O).$$

Since $L(O) < L(A)$, so $L(A, v_2) + \frac{Rx}{E(v_2)} < L(A)$—that is, $L(A, v_2) + \frac{Rx}{E(v_2)} < L(A, v_0)$. So the same as in case (a), P' is included in a cost-reducing path in R_f, which contradicts the definition of f.

Since P' is included in a cost-reducing path in R_f in both cases, the proof is complete.

From Theorem 2, we observe that our SubProb can be solved by finding a min-cost max-flow in G_N. Similar to Fakcharoenphol et al. [2010], we construct a min-cost max-flow as follows: from an arbitrary max flow in G_N, we need to cancel all of its cost-reducing paths to get the min-cost max-flow whose corresponding semi-matching has the minimum of the maximum load.

We can exploit the idea of Dinitz's blocking flow [Dinic 1970] to use a divide-and-conquer algorithm [Fakcharoenphol et al. 2010] to solve the general version of the semi-matching problem.

In Algorithm 1, C is divided into C_1 and C_2 of equal size in such a way that for any $c \in C_2$, the cost of edge (v, c) in G_N is greater than that of any edge adjacent to C_1.

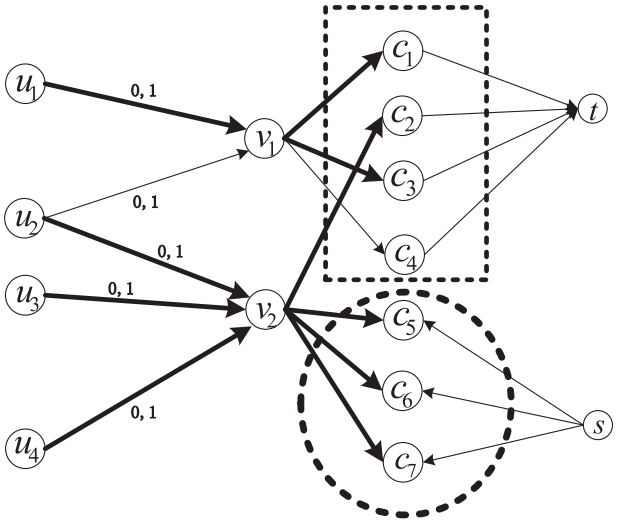

Fig. 4. Divide C into C_1 and C_2 of equal size. For any node $c \in C_2$, the cost of edge (v, c) for $v \in N_h$ is greater than that of edges adjacent to C_1.

ALGORITHM 1: CancelAll

Input: graph G_N. C are sorted in increasing order by the cost of edge (v, c), for $v \in N_h$ adjacent to them. There has existed a maximum flow f in G_N.

1: **if** $|C| = 1$ **then** halt **end if**.
2: divide C into C_1 and C_2 of equal size (showing in Figure 4, for any node $c \in C_2$, the cost of edge (v, c) for $v \in N_h$ is greater than that of edges adjacent to C_1).
3: Cancel(G_N, C_2, C_1).
4: $G_{N2} \leftarrow$ subgraph that is reachable from C_2.
5: $G_{N1} \leftarrow G_N - G_{N2}$.
6: recursively solve CancelAll(G_{N1}) and CancelAll(G_{N2}).

The function Cancel(G_N, C_2, C_1) cancels all cost-reducing paths from C_2 to C_1 in R_f by finding a maximum flow from C_2 to C_1 in R_f.

We assume that there is a dummy source s and a dummy sink t connecting to vertices in C_2 and in C_1. We iteratively use the Dinitz's blocking flow [Dinic 1970] to find a maximum flow. Since each vertex in N_{h+1} has in-degree 1 and R_f is a unit-capacity network, the total number of iterations is $O(\sqrt{|N_{h+1}|})$ [Fakcharoenphol et al. 2010], and each iteration of finding a blocking flow can be finished in $O(|E_h| + |N_h| + |N_{h+1}|)$ time. So the function Cancel(G_N, C_2, C_1) terminates in $O((|E_h| + |N_h| + |N_{h+1}|)\sqrt{|N_{h+1}|})$ time.

LEMMA 3 ([FAKCHAROENPHOL ET AL. 2010]). *The time complexity of Algorithm 1 is* $O((|E_h| + |N_h| + |N_{h+1}|)\sqrt{|N_{h+1}|}\log(|E_h| + |N_h|))$.

THEOREM 3. *Algorithm 2 finishes in* $O(|E|\sqrt{N}\log(N))$ *time.*

PROOF. From Lemma 3, we can use min-cost max-flow approach to get the optimal semi-matching for each height in the fat tree. The optimal shortest path tree is the union of these semi-matchings (Lines 4–12). Constructing a fat tree of the graph needs $O(N + |E|)$ running time. Initializing an arbitrary maximum flow needs $O(|E_h|)$ running time at each level. Since E_h is the edges in fat tree, we have

ALGORITHM 2: MLST

Input: graph $G(V, E)$, base station S. $E(i)$ for node i.
Output: maximum lifetime shortest path tree T_{opt}.

```
T ← ∅.
construct a fat tree for the graph.
d ← height of the fat tree.
for h = 1, ...,d − 1 do
    N_h ← set of the nodes at height h.
    N_{h+1} ← set of the nodes at height h + 1.
    G_h ← (N_{h+1} ∪ N_h, E_h).
    construct a network G_N from G_h.
    initialize G_N with an arbitrary maximum flow.
    CancelAll(G_N).
    T ← T ∪ { semi-matching in G_N}.
end for
T_opt ← T.
return T_opt.
```

$N = |V| = \sum_{h=1,2,\ldots,d} |N_h|$ and $|E| \geq \sum_{h=0,1,\ldots,d} |E_h|$. The overall complexity is

$$\begin{aligned}
&\sum_{h=1,2,\ldots,d} (|E_h| + |N_h| + |N_{h+1}|)\sqrt{|N_{h+1}|}\log(|E_h| + |N_h|) \\
&\quad \leq \sum_{h=1,2,\ldots,d} (|E_h| + |N_h| + |N_{h+1}|)\sqrt{N}\log(|E| + N) \\
&\quad \leq (|E| + 2N)\sqrt{N}\log(|E| + N) \\
&\quad = O(|E|\sqrt{N}\log N),
\end{aligned}$$

where the last equality comes from the fact that the graph is simple and connected so that $|E| = O(N^2)$ and $N = O(|E|)$. □

5. A DISTRIBUTED APPROACH

We have presented a centralized algorithm. In this section, we propose a distributed protocol. This protocol is based on the following observations. First, for nodes at different levels, their children assignments are independent from others. We can consider nodes at the same level and use their corresponding bipartite graph for children assignment. In fact, this is the basic idea to decompose MinST into SubProb. In addition, inner dependencies of WSNs can be revealed dynamically by some existing schemes (e.g. Liu et al. [2010]). Second, the bipartite graph may be disconnected, and each connected component has independent children assignment from other components. Therefore, we only need to consider how to assign children in each connected component $G_h = (N_{h+1} \cup N_h, E_h)$. (The distributed protocol also works if G_h is not connected.) This is a semi-matching problem that can be solved by Algorithm 1. Unfortunately, this algorithm is hard to modify to a distributed one. Therefore, we design a distributed algorithm to solve this problem by message passing.

5.1. Distributed Semi-Matching Algorithm

Our distributed semi-matching algorithm, Algorithm 3, is generalized from $\mathcal{A}_{SM1}$ [Harvey et al. 2003]. In the following, we will show that Algorithm 3 is correct.

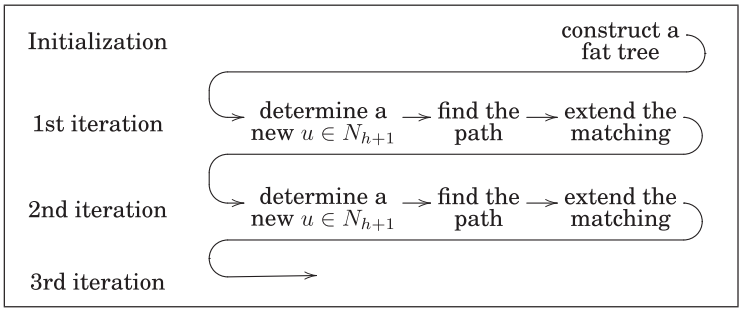

Fig. 5. Iterations of the approach.

ALGORITHM 3: Distributed Semi-Matching Algorithm

Input: bipartite graph $G_h = (N_{h+1} \cup N_h, E_h)$.
Output: The optimal semi-matching M.
1: $M \leftarrow \oslash$.
2: $S \leftarrow \oslash$.
3: **while** $S \neq N_{h+1}$ **do**
4: construct an alternating search tree T rooted at $u \in (N_{h+1} - S)$, where edges in M are directed from N_h to N_{h+1} and edges not in M are directed from N_{h+1} to N_h.
5: get a path $P = (u, \ldots, v)$ ($v \in N_h$ and v is a node in T) such that $\frac{Rx}{E(v)} \cdot (deg_M(v) + 1) + \frac{Tx}{E(v)}$ is minimum in the tree T.
6: $M \leftarrow M \bigoplus P$.
7: $S \leftarrow S \cup \{u\}$.
8: **end while**
9: return M.

LEMMA 4. *For the SubProb in a bipartite graph $G_h = (N_{h+1} \cup N_h, E_h)$, a semi-matching M is optimal if there is no alternating path $P = (u_1, v_1, \ldots, u_k, v_k)$ such that $(v_i, u_{i+1}) \in M$, $(u_i, v_i) \in (E_h - M)$ and $\frac{Rx}{E(v_1)} \cdot deg_A(v_1) + \frac{Tx}{E(v_1)} > \frac{Rx}{E(v_k)} \cdot (deg_A(v_k) + 1) + \frac{Tx}{E(v_k)}$.*

The proof is similar to that of Theorem 2, thus we omit it.

THEOREM 4. *Algorithm 3 finds an optimal semi-matching.*

PROOF. First, Algorithm 3 produces a semi-matching that has no alternating path defined in Lemma 4. The reason is similar to that of $\mathcal{A}_{SM1}$ [Harvey et al. 2003]. Then, according to Lemma 4, the semi-matching is optimal.

The core idea of Algorithm 3 is to iteratively add node $u \in N_{h+1}$ to the matching. After adding all nodes in N_{h+1}, the algorithm terminates and it guarantees that the achieved matching is optimal. We now design a distributed protocol based on Algorithm 3.

5.2. Distributed Protocol

We assume that nodes in the network are synchronized and that there is no packet loss in the transmission. Our distributed approach is based on the distributed DFS and BFS. The process of the protocol is to implement distributed DFS in G_h, and it mixes several iterations to add all nodes of N_{h+1} into the matching M. Each node $u \in N_{h+1}$ leads an iteration. We add a flag to indicate whether u has led an iteration.

If a node $u \in N_{h+1}$ has been visited by the distributed DFS, it suspends the distributed DFS and starts a new iteration. As is shown in Figure 5, each iteration consists of three

phases: determining a node for leading the iteration, finding the desired path, and extending the matching. First, u broadcasts a BFS message to construct an alternating search tree T rooted at itself in a distributed way. After the tree is constructed, each node in the tree waits to collect all replies from its children, then selects the optimal node from its children and itself. This process finds an optimal path directed from u and extends the matching along the path. After the iteration, u sets the flag to true and continues the distributed DFS by sending a distributed DFS message.

5.2.1. Initialization Phase. In this phase, the network invokes a distributed BFS to construct a fat tree rooted at the sink. When a node receives a BFS message for the first time, it records the source as its parent and then broadcasts a BFS message. Its height is the height of its parent plus 1. After receiving subsequent BFS messages, it updates its parent set and the children set under the fat tree. Then, for each corresponding connected bipartite graph at each level, nodes can implement the protocol independently. In this phase, each node u in the network transmits one message and receives d_u messages, where d_u is the degree of the node u in the fat tree.

5.2.2. Determining a Node. In this phase, the protocol determines a unique node in $(N_{h+1} - S)$ for leading the current iteration. S is the set of nodes in N_{h+1} that have led iterations. At the beginning of the protocol, S is empty. For the first iteration of the protocol in G_h, each node needs to flood in G_h. The node with the smallest ID in N_{h+1} leads the current iteration. For other iterations, the node u' leading the previous iteration is responsible for determining the leading node. This can be done by continuing distributed DFS from u'. Once a node $u \in (N_{h+1} - S)$ receives a distributed DFS message, u suspends distributed DFS and leads the current iteration. So the protocol is the process of implementing a distributed DFS that is mixed by several iterations. If it is the first iteration in G_h, each node broadcasts $|N_{h+1}|$ messages. For all other iterations, the whole overhead in this phase is to implement exactly a distributed DFS in G_h. Thus, each node transmits at most d_u messages and receives at most d_u messages.

5.2.3. Finding the Desired Path. This phase consists of two processes: constructing an alternating search tree and aggregating the desired path. Once a node u is determined in the last phase, u broadcasts a BFS message to construct a distributed alternating search tree T rooted at u, where edges in the matching are directed from N_h to N_{h+1} and edges not in the matching are directed from N_{h+1} to N_h. If a node receives a BFS message from u and finds that it is in T, it records the corresponding node as the parent in T and then broadcasts a BFS message. After that, it waits until it has collected all aggregation messages from its children in T. Then, it computes the optimal node v^* in $N_h \cap T$ with minimum value $\frac{Rx}{E(v^*)} \cdot (deg_M(v^*)+1) + \frac{Tx}{E(v^*)}$ in the subtree rooted at itself. It records its corresponding next hop and reports an aggregation message that contains the node v^* and the corresponding value to its parent. Finally, u gets the desired path to the destination node v^*. In this phase, each node transmits at most two messages and receives at most $2d_u - 1$ messages.

5.2.4. Extending the Matching. In this phase, nodes along the path switch the corresponding matching and nonmatching edges in the path. u starts this phase by sending a modification message to the next hop. When a node receives a modification message, it switches the state of edges in the path. After sending a modification message, u reactivates the distributed DFS by sending a DFS message. In this phase, each node transmits at most one message and receives at most one message.

The protocol runs $|N_{h+1}|$ iterations in G_h. In the protocol, each node in G_h transmits at most $3 \cdot |N_{h+1}| + d_u$ messages and receives at most $2 \cdot |N_{h+1}| \cdot d_u + d_u$ messages.

Table II. Configuration

Number of nodes	100 ∼ 1,000
Field (meter × meter)	100 × 100
Sink position	(50, 50)
Transmission range	20
Energy of each node	random (30, 50)
Energy consumption for transmission	1
Energy consumption for reception	1
Number of runs	10,000

Meanwhile, a node participates in at most two bipartite graphs. The total number of transmitted messages in the network is $\sum_u O(|N_{h+1}|+d_u)$ and that of received messages is $\sum_u O(|N_{h+1}| \cdot d_u)$. So the complexity in the worst case is $O(N|E|)$, which is the same as the time complexity of Algorithm 3. Although it is high in the worst case, we show in the later section that the overhead is very small in the distributed environment.

6. SIMULATION RESULTS

We evaluate the performance of our approaches through extensive simulations. Table II shows the simulation settings. Nodes are uniformly deployed in a 100m × 100m field with a sink located at $(50, 50)$. Two nodes can communicate with each other if the distance between them is less than the transmission range. According to CC2420's datasheets [Texas Instruments 2007], the energy consumption power for sending is quite close to receiving (17.4mA and 18.8mA, respectively). Other related works have similar results, which indicate that the energy consumption for sending and receiving a packet are almost the same. We therefore set $Tx/Rx = 1$ by default in simulations. The initial energy of each node is a random number in $[30, 50]$. Each node generates a message in a round. In the following subsection, we first present results on the performances of the centralized approach (MLST) and the impact of different network parameters. We compare our optimal scheme with two other related solutions. Last, we present the evaluation results of the distributed approach.

6.1. Performance of the Centralized Approach

This simulation contains three parts. In the first part, we deploy 500 nodes randomly and uniformly in the field and compute the optimal shortest path tree and its lifetime. We compare our optimal scheme with two other schemes: the random-case scheme and the worst-case scheme. In the random-case scheme, we randomly select a shortest path tree and compute its lifetime; in the worst-case scheme, we choose the shortest path tree with the minimum lifetime. The random scheme is widely used in many data collection protocols, and the worst scheme gives the lower bound of the network lifetime. We present detailed descriptions on these two schemes later. In the second part, we evaluate the algorithm performance with different network parameters. We compare the centralized approach with two other existing works as well in Section 6.1.3.

6.1.1. Lifetime Performance. We define the random scheme and the worst scheme. For the random scheme, each node randomly chooses a node as its parent from its candidate parent set. After getting the tree, we compute its lifetime. In the worst scheme, we select a shortest path tree with the minimum lifetime. Since every node has a candidate children set $cs(i)$ in the fat tree, we only need to find a node that has the minimum value $\frac{E(i)}{Tx+|cs(i)|Rx}$ and let this node choose all nodes of its candidate children set as children. Other nodes just arbitrarily choose their children. The result of this strategy is the shortest path tree with the minimum lifetime. We compare the lifetime of our

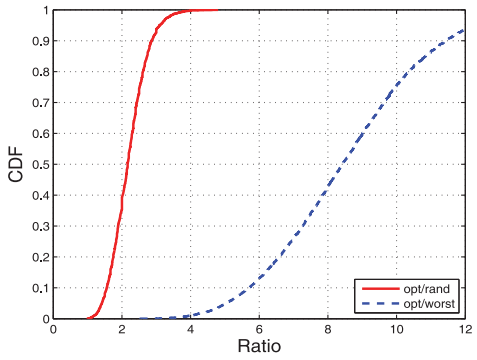

Fig. 6. Lifetime gain of our optimal scheme compared with the random/worst shortest path data aggregation tree.

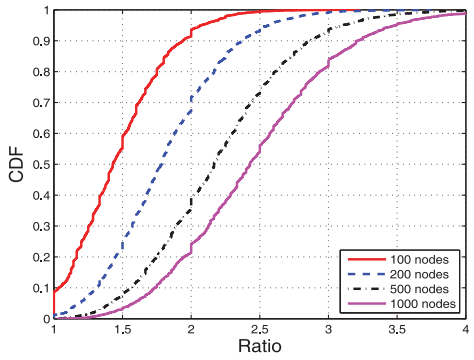

Fig. 7. Impact of the density on lifetime.

optimal scheme with these two schemes: 10,000 graphs are randomly generated, and for each we compute the ratio of the lifetime of the optimal shortest path tree to the ones of other two schemes.

The cumulative distribution function is plotted in Figure 6, where two curves start from ratio 1. As the solid curve shows, the lifetime of our optimal scheme outperforms the random shortest path tree with a median improvement of 210%. In some runs, our optimal scheme can prolong the lifetime by 4.5 times. When compared with the worst-case scheme (dashed curve in Figure 6), the lifetime ratio is much larger. Our optimal scheme has an improvement of 820% on average. The ratio is between 4 and 12 for most runs, and sometimes it even reaches 15, which is considerable.

6.1.2. Impact of Different Parameters. In this part, we analyze the impact of network parameters on the performance of our proposed centralized approach. Here we use the performance of random scheme as the baseline and compare the lifetime of our optimal scheme with it.

Effects of the node density. Node density of the network can affect the lifetime. We deploy different number of nodes in a 100m $\times$ 100m field—namely, 100 nodes, 200 nodes, 500 nodes, and 1,000 nodes, respectively. For each setting of node density, we run the simulation 10,000 times. As shown in Figure 7, our optimal scheme performs better when the node density is higher. When 100 nodes are deployed in the field, the lifetime gain ratios are in the range of 1 to 3.5 for most runs and the average value is 1.40. If the number of the nodes is 1,000, the ratios are in the range of 1 to 5, whereas the average ratio is 2.4. This can be explained as follows. When the node density increases, each node has more candidate parents, so the tree found by the optimal scheme will be more balanced from the perspective of energy consumption. We conclude that our optimal scheme is more effective when the network is dense.

Effects of initial energy. Different settings of initial energy on nodes might influence the performance of our proposed approach as well. We deploy 200 nodes in a 100m $\times$ 100m field with different ranges of initial energy: [1, 10], [1, 5], and [1, 2]. We run the simulation 10,000 times for different settings of initial energy. As shown in Figure 8, our optimal scheme performs better when nodes have more initial energy. When the initial energy is in [1, 2], in more than 95% of simulations, the lifetime gain ratios are in the range of 1 to 2.5. If the initial energy is in [1, 5], the ratios are in the range of 1 to 4 for most runs, and the ratios are in the range of 1 to 5 when the initial energy is in [1, 10]. This can be explained as follows. When the initial energy is distributed in

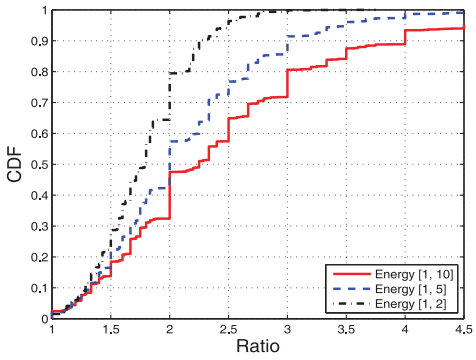

Fig. 8. Impact of the initial energy.

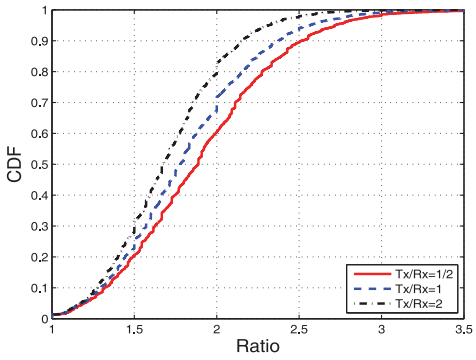

Fig. 9. Impact of the Tx/Rx.

a larger interval, the difference of initial energy increases among nodes so that assigning children according to initial energy has a larger impact on the network lifetime. We observe that our scheme works more effectively when the initial energy is much more unbalanced.

Effects of the relative energy consumption ratio of transmitting/receiving a packet. When considering network lifetime, we are interested in the impact of the relative difference of energy consumption between packet transmission and reception. We deploy 200 nodes in a 100m $\times$ 100m field. We compare the lifetime under different ratios of energy consumption of transmitting/receiving a packet. For each setting, we repeat the simulations 10,000 times. As shown in Figure 9, our optimal scheme performs better when Tx/Rx is small. If $Tx/Rx = 1/2$, about 40% of our optimal scheme performances are at least two times better, and only 20% of the optimal performances are two times better when Tx/Rx is 2. This is in accordance with our intuition. Considering the extreme case where receiving a message does not consume energy, then all spanning trees, including shortest path trees, will have the same lifetime. We conclude that our optimal scheme is more effective when the energy consumption of receiving a packet is relatively larger compared to the energy consumption of transmitting a packet.

Effects of transmission range. The transmission range of sensor nodes can influence the network lifetime. We evaluate the performance of our proposed approach under different settings of transmission range. We deploy 200 nodes in a 100m $\times$ 100m field. For each setting, the simulation is repeated 10,000 times. As shown in Figure 10, our optimal scheme performs better when the transmission range is larger. The average ratio is about 1.7 when the transmission range is 20m and is about 2.3 when the range is 30m. The reason is similar to that of increasing node density. With the increase of transmission range, the number of neighbors of a node increases so that a node may have more candidate parents.

6.1.3. Comparison with Two Related Solutions. In this section, we compare the performance of our optimal scheme with that of two other existing solutions in terms of network lifetime. One solution is an approximation algorithm proposed in Wu et al. [2008b]. The other is a data collection protocol in Baydere et al. [2006].

Compared to approximation algorithm. Wu et al. [2008b] give an approximation algorithm for finding a maximum lifetime data aggregation tree in WSNs. Note that the tree by this approximation algorithm is not restricted to shortest path tree. We randomly deploy 200 nodes in a 100m $\times$ 100m field. The transmission range is 20m, and the energy consumption for transmitting/receiving a packet is 2/1. One important

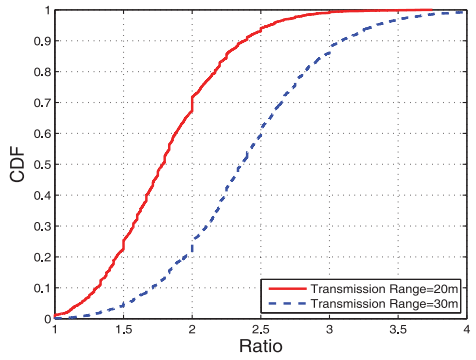

Fig. 10. Impact of the transmission range.

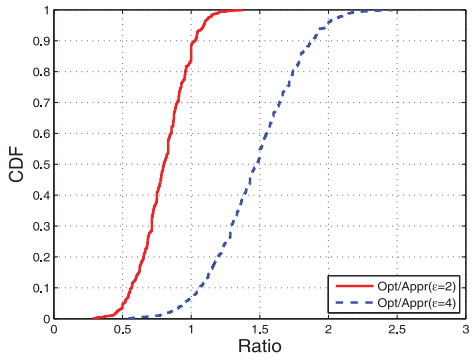

Fig. 11. Lifetime gain of our optimal scheme (Opt) compared with the approximation solution (Appr) in Wu et al. [2008b].

parameter of this algorithm is ϵ. A smaller ϵ means a longer network lifetime but higher time complexity. We set ϵ to 2 and 4, respectively, and run each simulation 10,000 times.

In Figure 11, we can see that in most cases (more than 90%), the lifetime of our solution is better than that of the approximation algorithm with $\epsilon = 4$. There are also 10% of cases where our solution is better than the approximation algorithm when $\epsilon = 2$. Note that although the approximation algorithm does not consider the data aggregation latency, we compare the performance in terms of data aggregation latency of these two schemes. The results are shown in Figure 12. Without loss of generality, we define the latency as the maximum distance from a sensor node to the sink. The latency in the approximation algorithm is at least 5 times of that in a shortest path tree in most situations. The network latency of our algorithm is 10 times better for more than 95% of cases when ϵ is 2 and for 85% cases when ϵ is 4. We can see that compared to the approximation algorithm proposed in Wu et al. [2008b], the latency of our scheme is much better and the lifetime is acceptable.

Compared to a data collection protocol. In this section, we compare the performance of our solution to the data collection protocol proposed in Baydere et al. [2006] in terms of network lifetime. Similar to ours, this protocol outputs shortest path trees. To compare our scheme with this protocol, we assume that nodes under both schemes can do in-network aggregation and that every node has a message to send in each round. We deploy 200 nodes in a 100m $\times$ 100m field. Each node has a transmission range of 20m. For a fair comparison, the lifetime is defined as the time duration until a fixed percentage of nodes die (or the network becomes disconnected). We consider four different definitions of lifetime: the time duration until the first node dies, until 1% of nodes die, until 2% of nodes die, and until 5% of nodes die. Note that our algorithm is specifically designed for the first lifetime definition. To adapt to the last three definitions of lifetime, we build a new tree based on the remaining nodes with residual energy whenever a node dies, until the total number of dead nodes reach the corresponding percentage. We show the lifetime gain in Figure 13. We have two observations. First, our algorithm is always better under the first lifetime definition due to the optimality of our algorithm under this definition. Second, under the other three definitions, our algorithm outperforms the data collection protocol in more than 95% of cases. The lifetime gain roughly increases with the increase of required percentage of dead nodes, because increasing the number of dead nodes will incur more executions of our algorithm with each execution returning an optimal tree (under the first definition) for the remaining nodes and energy. Note that our adapted solution is not always

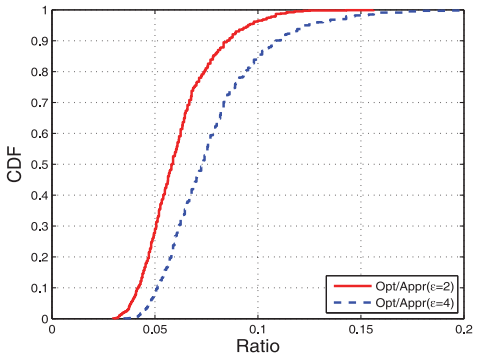

Fig. 12. Latency gain of our optimal scheme (Opt) compared to the approximation solution (Appr) proposed in Wu et al. [2008b].

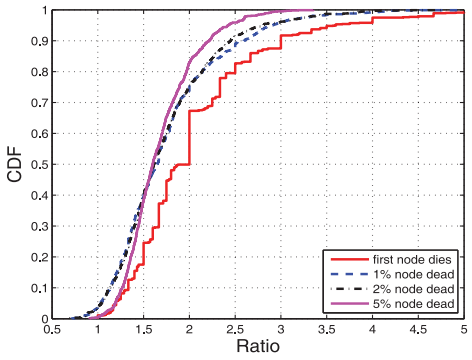

Fig. 13. Lifetime difference of our optimal scheme compared to the data collection protocol proposed in Baydere et al. [2006].

optimal under these three definitions of lifetime because the network topology would change after some nodes die. It is important to consider the set of dead nodes jointly. This is why, in some cases, the data collection protocol [Baydere et al. 2006] performs better.

6.2. Performance of the Distributed Protocol

In this subsection, we evaluate the performance of the distributed protocol. Nodes are randomly deployed in the regular field uniformly. We design two sets of experiments and run each set 500 times. For the first set, we increase the number of nodes while keeping the node density, which means that we change the deployment area proportionally. For the second set, we fix the area of the field and increase the number of nodes deployed. In both sets of experiments, we study the convergence time and each node's overhead, which is measured by the number of messages a node transmits during the protocol.

6.2.1. Performance of the Protocol with Fixed Node Density. We first fix the node density in the simulation. We study the convergence time, which is represented by the number of iterations a node takes part in the protocol. As Figure 14 shows, the average convergence time (dashed line) of each node is about 20 iterations and slowly increases with the number of nodes. The value is 18.48 when the number of nodes is 100 and reaches 24.5 for 600 nodes. Thus, each node is involved in only a few iterations in the distributed protocol. The maximum number of iterations (solid line) among nodes in the network also increases slowly. The value increases from 29.5 to 56.8, whereas the number of nodes increases from 100 to 600.

We then evaluate the performance in terms of the overhead. We use the number of messages that a sensor transmits as its overhead during the protocol. As Figure 15 shows, each node transmits 60 messages on average (dashed line) regardless of the number of nodes in the network. It is much smaller than the worst case, which is $O(N)$. The maximum overhead (solid line) of nodes in the network is about two to three times the average overhead.

The simulation results show that the overhead of the network is tolerable, and thus our distributed approach works pretty well.

6.2.2. Performance of the Protocol with Variable Node Density. We now study the impact of the node density on the performance of the distributed protocol. We vary the number of nodes from 100 to 600 at a step of 100.

Figure 16 shows that the convergence time (dashed line) is closely related to the node density. When there are 100 nodes in the network, a node takes 18.48 iterations on average. It increases almost linearly to 94.9 iterations when the number of nodes

60
55
50
45
40
35
30
25
20
15
100
200
300
400
500
600
Number of nodes

are some related works focusing on the load balance problems in data collection sensor networks, such as Imon et al. [2013], which aims at maximizing the lifetime of WSNs through load balancing. We show that the resulting problem is not only NP-hard but also APX-hard. We give the decision version of the problem as follows.

Definition 1 *(MLSToA).* Suppose that nodes cannot do aggregation. Given graph $G(V, E)$, each nonroot node i's initial energy $E(i)$ for $i = 1, 2, \ldots, N$, two positive numbers Tx and Rx for energy consumption of transmission and receiving, respectively, and a positive number L, does there exist a shortest path tree T with a lifetime at least L?

Note that without aggregation, node i needs to receive $D(T, i)$ messages and transmit $D(T, i)+1$ messages in each round where $D(T, i)$ is the number of node i's descendants in tree T, whereas in the aggregation model, it needs to receive $C(T, i)$ messages and transmit one message in each round where $C(T, i)$ is the number of node i's children in tree T. This is the only difference between MLST and the maximum lifetime shortest path tree without aggregation (MLSToA).

THEOREM 5. *MLSToA is NP-complete.*

PROOF. First, MLSToA is in NP. To see this, construct a fat tree as before. Then, nondeterministically select a shortest path tree from the fat tree, and check whether this selected tree has lifetime at least L. Obviously, both the selection and checking can be done in polynomial time.

Second, MLSToA is NP-hard. This can be proved by reducing from the NP-hard problem, three-dimensional matching [Gary and Johnson 1979].

Definition 2 *(3DM).* Given set $M \subseteq A \times B \times C$ where $A = \{a_1, a_2, \ldots, a_p\}$, $B = \{b_1, b_2, \ldots, b_p\}$, and $C = \{c_1, c_2, \ldots, c_p\}$ are disjoint sets having the same number p of elements, does there exist a matching $M' \subseteq M$ such that $|M'| = p$ and no two elements of M' agree in any coordinate?

Given a 3DM instance, construct an MLSToA instance as follows. We may assume $|M| \geq p$; otherwise, the answer to 3DM is trivially negative, and we can construct a trivial MLSToA instance that gives negative answer. Set $Tx = 0$ and $Rx = 1$. The constructed graph $G(V, E)$ is a fat tree rooted at the sink. At height 0, there is one node—the sink—with infinity energy. At height 1, there are $|M|$ nodes, each of which has initial energy of 3 and corresponds to a distinct matching in M. They are the children of the sink. At height 2, there are $3p$ normal nodes and $|M| - p$ "dummy" nodes. Each of these nodes has initial energy of 4. A normal node i corresponds to an element in $A \bigcup B \bigcup C$, and it is adjacent to a node at height 1 if and only if this node i appears in the corresponding matching. On the other hand, a dummy node is adjacent to all nodes at height 1, and any dummy node has two distinct children at height 3, each of which has initial energy 4. The purpose of dummy nodes is to occupy $|M| - p$ nodes at height 1 so that there are p nodes left at height 1 corresponding to the selected matching M'. In summary, there are $1 + 4|M|$ nodes in the graph: 1 node at height 0, $|M|$ nodes at height 1, $2p + |M|$ nodes at height 2, and $2(|M| - p)$ nodes at height 3. Similarly, there are $|M| + (3|M| + (|M| - p)|M|) + 2(|M| - p)$ edges in the graph. Therefore, the length of the constructed MLSToA instance is polynomial in the input length of 3DM so that the reduction can be done in polynomial time. Under our construction, different shortest path trees only differ in their selection of edges connecting nodes at height 1 with nodes at height 2.

It is straightforward to see that there is a matching M' for 3DM if and only if the corresponding MLSToA has a shortest path tree T with a lifetime of at least 1. Suppose

that there exists a matching M', then construct T as follows. Assign each node in $M-M'$ a distinct dummy node as child, and assign each node in M' the three corresponding nodes at height 2 as children. Then, T has a lifetime of 1. On the other hand, suppose that there exists a shortest path tree T with a lifetime of at least 1. Under T, each node at height 1 can at most have one dummy node as a child, so there are exactly p nodes at height 1 not having dummy nodes as child. These p nodes are the elements of M'. We can check that M' is the desired matching. □

In fact, the proof can yield a stronger result that the maximization version of MLSToA is APX-hard. Specifically, we have the following corollary.

COROLLARY 1. *If nodes cannot perform aggregation, then there is no polynomial time algorithm that can construct a shortest path tree with lifetime larger than* 3/4 *of the maximum lifetime, unless* $P = NP$.

PROOF. We prove the corollary by showing that any such algorithm can be used to solve 3DM. Recall the constructed MLSToA instance in the proof of Theorem 5. We can check that there is a matching for 3DM if and only if the constructed MLSToA has a shortest path tree with lifetime larger than 3/4. (Lifetime larger than 3/4 is actually equivalent to lifetime of at least 1 for the constructed MLSToA.) In addition, the maximum lifetime of the MLSToA is 1. Thus, the aforementioned algorithm can answer the latter question so that it can solve 3DM. □

The preceding results show that aggregation capability is critical in producing polynomial time algorithm in our work. Without this capability, it is difficult to find a shortest path tree with maximum lifetime.

8. CONCLUSIONS

In this article, we have studied the problem of finding a shortest path tree with the maximum lifetime when in-network aggregation is used. We have transformed the problem into the load balancing scheme at each level of the fat tree, proved that the problem is in P, and solved it by a centralized approach in polynomial time. A distributed approach has also been proposed for better applicability. We verify our results through extensive simulations. We also extend our discussion to WSNs without aggregation and present interesting theoretical results.

REFERENCES

Sebnem Baydere, Yasar Safkan, and Ozlem Durmaz. 2006. Lifetime analysis of reliable wireless sensor networks. *IEICE Transactions on Communications* 88, 6, 2465–2472.

Guihai Chen, Chengfa Li, Mao Ye, and Jie Wu. 2009. An unequal cluster-based routing protocol in wireless sensor networks. *Wireless Networks* 15, 2, 193–207.

Xujin Chen, Xiaodong Hu, and Jianming Zhu. 2005. Minimum data aggregation time problem in wireless sensor networks. In *Proceedings of the 1st International Conference on Mobile Ad-hoc and Sensor Networks*. Springer-Verlag, Berlin, Heidelberg, 133–142.

Efim A. Dinic. 1970. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Mathematics–Doklady* 11, 1277–1280.

Jittat Fakcharoenphol, Bundit Laekhanukit, and Danupon Nanongkai. 2010. Faster algorithms for semi-matching problems. In *Proceedings of the 37th International Colloquium Conference on Automata, Languages and Programming (ICALP'10)*. Springer-Verlag, Berlin, Heidelberg, 176–187.

Michael R. Gary and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Macmillan Higher Education.

Arvind Giridhar and Praveen R. Kumar. 2005. Computing and communicating functions over sensor networks. *IEEE Journal on Selected Areas in Communications* 23, 4, 755–764.

Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. 2009. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM, New York, NY, 1–14.

Nicholas J. A. Harvey, Richard E. Ladner, László Lovász, and Tami Tamir. 2003. Semi-matchings for bipartite graphs and load balancing. In *Algorithms and Data Structures*. Lecture Notes in Computer Science, Vol. 2748. Springer, 294–306.

Scott C.-H. Huang, Peng-Jun Wan, Chinh T. Vu, Yingshu Li, and Frances Yao. 2007. Nearly constant approximation for data aggregation scheduling in wireless sensor networks. In *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM'07)*. IEEE, Los Alamitos, CA, 366–372.

Sk Kajal Arefin Imon, Adnan Khan, Mario Di Francesco, and Sajal K. Das. 2013. RaSMaLai: A randomized switching algorithm for maximizing lifetime in tree-based wireless sensor networks. In *Proceedings of the 32nd IEEE International Conference on Computer Communications (INFOCOM'13)*. IEEE, Los Alamitos, CA, 2913–2921.

Texas Instruments. 2007. CC2420 datasheet. Available at http://www.ti.com/product/cc2420.

Tung-Wei Kuo and Ming-Jer Tsai. 2012. On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: NP-completeness and approximation algorithms. In *Proceedings of the 31st International Conference on Computer Communications (INFOCOM'12)*. IEEE, Los Alamitos, CA, 2591–2595.

Hieu Khac Le, Dan Henriksson, and Tarek Abdelzaher. 2007. A control theory approach to throughput optimization in multi-channel collection sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*. ACM, New York, NY, 31–40.

Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, and Thomas H. Cormen. 2001. *Introduction to Algorithms*. MIT Press.

Chieh-Jan Mike Liang, Jie Liu, Liqian Luo, Andreas Terzis, and Feng Zhao. 2009. Racnet: A high-fidelity data center sensing network. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM, New York, NY, 15–28.

Yunhao Liu, Kebin Liu, and Mo Li. 2010. Passive diagnosis for wireless sensor networks. *IEEE/ACM Transactions on Networking* 18, 4, 1132–1144.

Yunhao Liu, Xufei Mao, Yuan He, Kebin Liu, Wei Gong, and Jiliang Wang. 2013. CitySee: Not only a wireless sensor network. *IEEE Network* 27, 5, 42–47.

Dijun Luo, Xiaojun Zhu, Xiaobing Wu, and Guihai Chen. 2011. Maximizing lifetime for the shortest path aggregation tree in wireless sensor networks. In *Proceedings of the 30th International Conference on Computer Communications (INFOCOM'11)*. IEEE, Los Alamitos, CA, 1566–1574.

Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. 2002. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*. ACM, New York, NY, 88–97.

Joongseok Park and Sartaj Sahni. 2006. An online heuristic for maximum lifetime routing in wireless sensor networks. *IEEE Transactions on Computers* 55, 8, 1048–1056.

Yi Shi and Yiwei Thomas Hou. 2008. Theoretical results on base station movement problem for sensor network. In *Proceedings of the 27th International Conference on Computer Communications (INFOCOM'08)*. IEEE, Los Alamitos, CA, 1–5.

Hüseyin Özgür Tan and Ibrahim Körpeoğlu. 2003. Power efficient data gathering and aggregation in wireless sensor networks. *ACM Sigmod Record* 32, 4, 66–71.

Peng-Jun Wan, Scott C.-H. Huang, Lixin Wang, Zhiyuan Wan, and Xiaohua Jia. 2009. Minimum-latency aggregation scheduling in multihop wireless networks. In *Proceedings of the 10th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, New York, NY, 185–194.

Xiaobing Wu, Guihai Chen, and Sajal K. Das. 2008a. Avoiding energy holes in wireless sensor networks with nonuniform node distribution. *IEEE Transactions on Parallel and Distributed Systems* 19, 5, 710–720.

Yan Wu, Sonia Fahmy, and Ness B. Shroff. 2008b. On the construction of a maximum-lifetime data gathering tree in sensor networks: NP-completeness and approximation algorithm. In *Proceedings of the 27th International Conference on Computer Communications (INFOCOM'08)*. IEEE, Los Alamitos, CA, 356–360.

Yanwei Wu, Xiang-Yang Li, YunHao Liu, and Wei Lou. 2010. Energy-efficient wake-up scheduling for data collection and aggregation. *IEEE Transactions on Parallel and Distributed Systems* 21, 2, 275–287.

Yafeng Wu, John A. Stankovic, Tian He, and Shan Lin. 2008c. Realistic and efficient multi-channel communications in wireless sensor networks. In *Proceedings of the 27th International Conference on Computer Communications (INFOCOM'08)*. IEEE, Los Alamitos, CA, 1193–1201.

Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. 2004. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. ACM, New York, NY, 13–24.

Yuan Xue, Yi Cui, and Klara Nahrstedt. 2005. Maximizing lifetime for data aggregation in wireless sensor networks. *Mobile Networks and Applications* 10, 6, 853–864.

Mao Ye, Chengfa Li, Guihai Chen, and Jie Wu. 2005. EECS: An energy efficient clustering scheme in wireless sensor networks. In *Proceedings of the 24th IEEE International Conference on Performance, Computing, and Communications (IPCCC'05).* IEEE, Los Alamitos, CA, 535–540.

Received August 2012; revised December 2012; accepted January 2014