This paper was presented as part of the main technical program at IEEE INFOCOM 2011

Maximizing Lifetime for the Shortest Path Aggregation Tree in Wireless Sensor Networks

Dijun Luo, Xiaojun Zhu, Xiaobing Wu and Guihai Chen
State Key Laboratory for Novel Software Technology
Nanjing University, Nanjing, P. R. China
Email: {luodijun, gxjzhu}@gmail.com, {wuxb, gchen}@nju.edu.cn

***Abstract*—In many applications of wireless sensor networks, a sensor node senses the environment to get data and delivers them to the sink via a single hop or multi-hop path. Many systems use a tree rooted at the sink as the underlying routing structure. Since the sensor node is energy constrained, how to construct a good tree to prolong the lifetime of the network is an important problem. We consider this problem under the scenario where nodes have different initial energy, and they can do in-network aggregation. In previous works, it has been proved that finding a maximum lifetime tree from all feasible spanning trees is NP-complete. Since delay is also an important element in time-critical applications, and shortest path trees intuitively have short delay, it is imperative to find a shortest path tree with long lifetime. This paper studies the problem of maximizing the lifetime of data aggregation trees, which are limited to shortest path trees. We find that when it is restricted to shortest path trees, the original problem is in P. We transform the problem into a general version of semi-matching problem, and show that the problem can be solved by min-cost max-flow approach in polynomial time. Also we design a distributed solution. Simulation results show that our approach greatly improves the lifetime of the network and is more competitive when it is applied in a dense network.**

I. Introduction

In many Wireless Sensor Networks(WSN) applications such as structure maintenance [1] and habitat monitoring [2], data collection is a basic operation: a sensor node senses the environment to generate data and sends them via a single hop or multi-hop path to the sink. During the data collection process, some divisible functions [3] (e.g. SUM, MAX, MIN, AVERAGE, top-k, etc.) can fuse data from different sensors to eliminate redundant transmissions. In order to collect data, many previous works use a routing tree structure (eg. [4], [5]) to collect data: a sensor receives data messages from its children and forwards them to its parent. Meanwhile, the sensor also transmits its own data to the parent. Since wireless sensor nodes are battery-powered and usually deployed in severe environments, it is impractical to replace the battery for the sensor node. So how to construct an efficient tree to prolong the lifetime of the network is an important problem.

Wu et al. [6] proved that finding a maximum lifetime arbitrary tree is NP-complete, and they proposed an approximation algorithm that produces a sub-optimal tree. Although the solution is "near optimal", it may result in long delay if the tree is deep. Consequently, it is not suitable for time-critical applications. A shortest path tree is a tree in which each node has the least hop count path to the sink. Intuitively, shorter path usually has shorter delay. It is important to find an optimal shortest path tree with maximum lifetime.

We study the problem of selecting a maximum lifetime tree from the set of shortest path trees. The problem can be reduced to a general version of *semi-matching* problem [7], [8]. We prove that it can be solved in polynomial time. Our contributions are enumerated as follows:

- 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 tree 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. To the best of our knowledge, this algorithm is the fastest in the literature. We first divide the problem into several sub-problems, each of which is a general version of a semi-matching problem. Since the existing solutions to the semi-matching problem can not be applied directly to the general version, we make modifications on the existing solutions to solve each sub-problem.
- For better applicability of our result to WSN, we present a distributed solution. Since it is hard to transform the proposed centralized algorithm to a distributed one, we use another algorithm for the semi-matching problem which has slightly high time complexity but is more suitable for distributed implementation.
- We carry out extensive simulations to verify our approaches. The results show that our approach greatly improves the lifetime of the network. We numerically analyze the distributed approach.

The rest of this paper is organized as follows. Section II reviews some related works on data collection and routing tree structure. Section III presents the model and formulation of the focused problem. We propose the solution and the centralized algorithm in Section IV. Section V presents our distributed approach. The simulation results are given in Section VI. Finally, we conclude our work in section VII.

II. Related Works

There are many related works for the problem of data gathering and aggregating from wireless sensor networks. In this section, we review the most related works.

Some works are based on non-tree routing structures [10], [11]. Park and Sahni [10] proved that maximum lifetime routing in wireless sensor network is NP-hard and proposed a heuristic approach. They did not consider the energy consumption for receiving messages. Shi et al. [11] took advantage of mobile sink for data collection to prolong the network lifetime. However, sometimes we may not afford to deploy mobile sinks.

Many data gathering and aggregation protocols are based on tree routing structures [4], [5], [12]. A well-known protocol is CTP [4], which computes anycast routes to sinks. Liang et al. [5] defined the length of the edge as the *expected transmission count (ETX)* and constructed a shortest path tree for each channel. Le et al. [12] proposed an approach to construct a shortest path tree for each sink and dynamically adjust to balance the load among the sinks. These works are based on real systems, but they do not provide theoretical insight into maximization of lifetime.

Recently, there are some works about constructing a tree routing structure with maximum lifetime. Xue et al. [13] used a linear programming approach to give an approximation algorithm for finding a maximum lifetime aggregation tree. Wu et al. [6] proved that constructing an arbitrary aggregation tree with the maximum lifetime is NP-hard, and proposed 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 [5], [12], [14]. So it is important to find an optimal shortest path tree. In this paper, we study the problem of finding a shortest path aggregation tree with maximum lifetime. We formulate the problem as a general version of *semi-matching* problem [7], [8] and solve it by min-cost max-flow approach [7].

III. Network Model and Problem Formulation

In this paper, we only consider the energy depletion in transmitting and receiving messages, which are the two major energy consuming operations [6]. In this section, we first present the model and the assumption of the network, and then formulate the problem a load balancing problem.

A. A Motivating Example

In this subsection, we present a simple example to show the importance of finding an optimal shortest path tree to prolong the lifetime of the network.

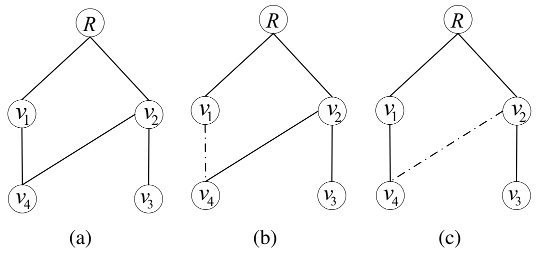

Fig. 1. A simple example for the optimal shortest path tree.

Figure 1(a) shows a small wireless sensor network. There are 5 nodes and 5 edges, and each node can do in-network aggregation. We assume that the root node R has infinite energy and non-root nodes (v_1, v_2, v_3, v_4) have initial energy $(2, 7, 3, 3)$ respectively. Each transmission and reception consumes 1 unit of energy. There are two shortest path trees as shown in Figures 1(b) and 1(c) (each solid line represents a tree edge).

In Figure 1(b), v_3 and v_4 are the children of v_2. In each round, v_2 receives two messages from v_3 and v_4, aggregates them with its local data 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 2 rounds, node v_1 depletes its energy and dies. So the lifetime of the network is 2 (represented by the number of rounds). In Figure 1(c), v_4 is the child of v_1 instead of v_2. In one round the energy consumption of (v_1, v_2, v_3, v_4) is $(2, 2, 1, 1)$, so the network's lifetime is 1. In comparison, we prefer the shortest path tree in Figure 1(b) to the one in Figure 1(c).

B. Network Model

There are N sensor nodes $(1, 2, \ldots, N)$ and a sink 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 non-replenishable energy $E(i)$ ($E(i) \geq 0$), while 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 multi-frequency 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. In this paper, we assume the routing structure is a static tree. Each node can do in-network aggregation: a node aggregates the received messages, 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 paper in Table I. In each round, a node generates one message and transmits it to the sink via a single hop or multi-hop 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 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

TABLE I
TERMINOLOGY

Symbol	Definition
$G(V,E)$	The graph of the network, V represents the set of nodes, while E corresponds to the set of edges.
T	The aggregation routing tree which is a shortest path tree and the sink is the root.
N	The number of nodes in the network, $N = \|V\|$.
Tx	Energy consumption for transmitting a message.
Rx	Energy consumption for receiving a message.
$E(i)$	The non-replenishable energy for node i, $i = 1, 2, \ldots, N$, $E(0) = \infty$ for the sink.
$C(T,i)$	The number of children for the node i in tree T.
$l(T,i)$	The lifetime of the node i in a routing tree T.
$l(T)$	The lifetime of a routing tree T, i.e., $l(T) = \min_{i=1,\ldots,N} l(T,i)$.
$L(T,i)$	The load of the node i in a routing tree T, defined as $L(T,i) = \frac{1}{l(T,i)}$.
$L(T)$	The load of the network under a routing tree T, i.e., $L(T) = \max_{i=1,\ldots,N} L(T,i)$.
$h(i)$	The minimum number of hops from the node i to the sink, $h(0) = 0$.
d	The maximum value of $h(i)$,which is the height of the fat tree.
L_h	The minimum of the maximum load for the nodes at height h.
SPT	The set of the shortest path trees in the network.
N_h	The set of nodes at height h.
E_h	The set of edges between nodes at height h and $h+1$.

literature (e.g., [6]). We have the lifetime of the network under a routing tree T

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

C. Problem Formulation

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

Problem 1 (MLST). *Given a graph $G(V,E)$, root R, and each node i's initial energy $E(i)$, $i = 0, 1, 2, \ldots, N$, the problem is to find a Maximum Lifetime Shortest path Tree T_{opt} from the set of shortest path trees in the graph. We call it MLST problem for short.*

Formally, we have

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

It is hard to directly solve this problem. So we transform it into an equivalent formulation. We 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 we 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 $G(V,E)$, root R, and each node i's initial energy $E(i)$, $i = 0, 1, 2, \ldots, N$, the problem is to find the Minimum load Shortest path Tree T'_{opt} among all the shortest path trees.*

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

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. ■

In the next section, we will solve the equivalent problem for the *MSLT* problem.

IV. A CENTRALIZED APPROACH

In order to solve the *MinST* problem, we first construct a fat tree [14] from the network and show that every edge of the 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* [8]. We prove that this general version of *semi-matching* can be solved by min-cost max-flow approach in polynomial time.

A. Fat Tree

We apply a Breadth-First search algorithm [16] to construct a fat tree rooted at the sink. Each node stores its height (hop counts) to the sink and 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, since it will never appear in any shortest path tree.

As Figure 2(a) shows, 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. We can remove edges (v_1, v_2) and (u_2, u_3) (dashed line) from the graph. So by using fat tree, we can reduce the number of edges to simplify the graph.

In order to get the optimal tree, each node needs to select several neighbors from its neighbors as its children.

B. Children Assignment and Load Balancing

Our goal is to find a minimum load shortest path tree 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, $C(T,i)$ and $E(i)$ determine the value of $L(T,i)$. So the load of a node is only affected by its number of children and its initial energy. We find that the children assignment for a node only affects the assignments for some special nodes in the network.

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

Proof: From equation (3), the value of $L(T,i)$ is only determined by $C(T,i)$ and $E(i)$. Each node's height equals its parent's height plus 1. So 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 $N_h - \{i\}$. So

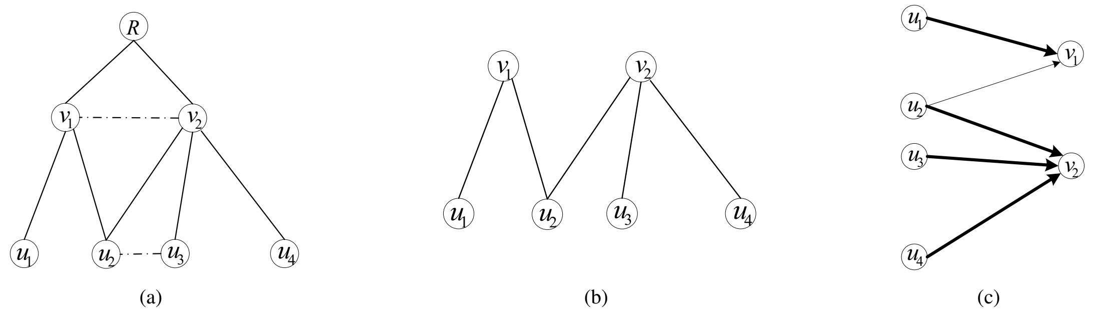

Fig. 2. (a) The fat tree of the graph. Edge(v_1, v_2) does not appear in any shortest path tree. (b) The nodes with same height in the fat tree. v_1 and v_2 have the same height. Their candidate children are also at the same height. (c) The general version of semi-matching. Each vertex u is incident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

we can do children assignment independently at each height. ■

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 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 above analysis, the *MinST* problem can be divided into subproblems.

Problem 3 (SubProblem)**.** *Given a bipartite graph $G_h = (N_{h+1} \cup N_h, E_h)$ (see 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. Let $deg_M(v)$ be the number of edges in M that are incident with $v \in N_h$, the load of $v \in N_h$ in M is*

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

The problem is to find an optimal semi-matching for G_h so that the maximum load in N_h is minimized.

Therefore we only need to solve the *SubProblem* and combine the solutions of all *SubProblems*, then we get the solution of *MinST* problem. Note that it is different from the definition of *semi-matching* in [8], 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 *SubProblem* is simplified to $deg_M(v)$. So our *SubProblem* is a general version of the *semi-matching* problem.

C. Solution

It has been proved in [7] that the problem of finding an optimal semi-matching M that has minimum of the maximum load $\max_v deg_M(v)$ can be solved by min-cost max-flow approach in polynomial time. In this subsection, we will exploit this approach and prove that our *SubProblem* can be solved in polynomial time.

Similar to [7], we show how to construct a network G_N such that a min-cost max-flow determines an optimal semi-matching in *SubProblem*.

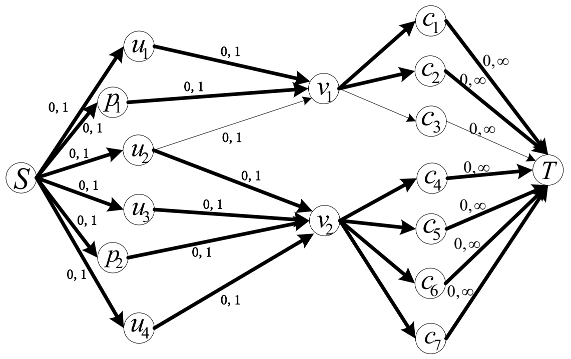

Fig. 3. The corresponding min-cost max-flow problem. Some edges are labeled with an entry (cost, capacity). Bold edges have flow of unit 1.

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 (see 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 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 $\{(S,u)|u \in N_{h+1}\cup P\}$ 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 [16], 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 ideas from [8] and have 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 [7].

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 $R_f(G_N)$.*

The process of the proof is presented in [7], 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* which 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 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. So 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 $1 + deg_A(v_2) \leq deg_O(v_2)$. So 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 $L(A,v_2) + \dfrac{Rx}{E(v_2)} < L(A,v_0)$. So the same as 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 *SubProblem* can be solved by finding a min-cost max-flow in G_N. Similar to [7], we find 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 [15] to use divide-and-conquer algorithm [7] to solve the general version of 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. 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.

Algorithm 1 CancelAll [7]

```
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 (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 which is reachable from $C_2$.
5:  $G_{N1} \leftarrow G_N - G_{N2}$.
6:  recursively solve CancelAll($G_{N1}$) and CancelAll($G_{N2}$).
```

We iteratively use the Dinitz's blocking flow [15] to find a maximum flow. Since each vertex in N_{h+1} has indegree 1 and R_f is a unit-capacity network, the total number of iterations is $O(\sqrt{|N_{h+1}|})$ [7], 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 ([7])**.** *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 (Line4-12). Constructing a fat tree of the graph needs $O(N + |E|)$ running time. Initializing a arbitrary maximum flow needs $O(|E_h|)$ running time at each level. Since E_h is the edges in fat tree, we have $N = |V| = \sum_{h=1,2,\ldots,d}|N_h|$ 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 $N = O(|E|)$. ■

V. A Distributed Approach

We can compute children assignment locally. First, for nodes at different levels, their children assignments are independent from each other. For nodes at the same level, we consider the corresponding bipartite graph for their children assignment. It may be disconnected. For every connected component, the children assignment is independent of that in other components. Therefore, we only need to consider how to assign children in each connected component $G_{h^i} = (N_{h+1^i} \cup N_{h^i}, E_{h^i})$. This is a semi-matching problem, which can be solved by Algorithm 1. Unfortunately, this algorithm is hard to be modified to a distributed one. Therefore, we choose another centralized algorithm $\mathcal{A}_{SM1}$ [8], generalize it, and design a distributed version.

Algorithm 2 *MLST*

```
Input: graph $G(V,E)$,base station $S$. $E(i)$ for node $i$.
Output: maximum lifetime shortest path tree $T_{opt}$.
1:   $T \leftarrow \oslash$.
2:   construct a fat tree for the graph.
3:   d $\leftarrow$ height of the fat tree.
4:   for $h = 1,\ldots,d-1$ do
5:       $N_h \leftarrow$ set of the nodes at height $h$.
6:       $N_{h+1} \leftarrow$ set of the nodes at height $h+1$.
7:       $G_h \leftarrow (N_{h+1} \cup N_h, E_h)$.
8:       construct a network $G_N$ from $G_h$.
9:       initialize $G_N$ with a arbitrary maximum flow.
10:      CancelAll($G_N$).
11:      $T \leftarrow T \cup \{$ semi-matching in $G_N\}$.
12:  end for
13:  $T_{opt} \leftarrow T$.
14:  return $T_{opt}$ .
```

Algorithm 3 General version of $\mathcal{A}_{SM1}$

```
Input: bipartite graph $G_{h^i} = (N_{h+1^i} \cup N_{h^i}, E_{h^i})$.
Output: The optimal semi-matching $M$.
1:  $M \leftarrow \oslash$.
2:  $S \leftarrow \oslash$.
3:  while $S \neq N_{h+1^i}$ do
4:      construct an alternating search tree $T$ rooted at $u \in (N_{h+1^i} - S)$,
        where edges in $M$ are directed from $N_{h^i}$ to $N_{h+1^i}$ and edges not
        in $M$ are directed from $N_{h+1^i}$ to $N_{h^i}$.
5:      get a path $P = (u,\ldots,v)$ ($v \in N_{h^i}$ 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 $M$.
```

A. Generalized $\mathcal{A}_{SM1}$

Algorithm 3 is the generalized $\mathcal{A}_{SM1}$. To see this, we can obtain $\mathcal{A}_{SM1}$ by setting $Tx = 0$ and $E(v) = Rx$ for all $v \in N_h$ (Line 5). In the following, we will show that Algorithm 3 is correct.

Lemma 4. *For the SubProblem in a bipartite graph* $G_{h^i} = (N_{h+1^i} \cup N_{h^i}, E_{h^i})$, *a semi-matching A is optimal if there is no alternating path* $P = (v_1, u_1, \ldots, v_k)$ *such that* $(v_i, u_i) \in A$, $(u_i, v_{i+1}) \in (E_{h^i} - A)$ *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. Due to limitation of space, we omit the proof here.

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 $\mathcal{A}_{SM1}$ [8]. Then, according to Lemma 4, it is optimal. ■

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

B. Distributed Protocol

We assume that nodes in the network are synchronized and 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^i} and it mixes several iterations to add all nodes of N_{h+1^i} into the matching M. Each node $u \in N_{h+1^i}$ leads an iteration. We add a flag to indicate whether u has led an iteration.

If a node $u \in N_{h+1^i}$ has been visited by the distributed DFS, it suspends the distributed DFS and starts a new iteration. As Figure 4 shows, 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, and 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.

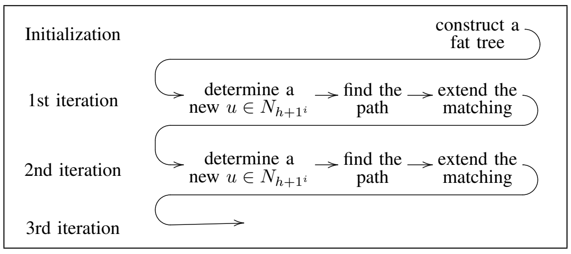

Fig. 4. Iterations of the approach.

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 first BFS message, 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 subsequence BFS messages, it updates its parent set and 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 1 message and receives d_u messages where d_u is the degree of the node u in the fat tree.

2) Determining a node u: In this phase, the protocol determines a unique node in $(N^i_{h+1} - S)$ for leading the current iteration. S is the set of nodes in N_{h+1^i} that have led iterations. At the beginning of the protocol, S is empty. For the first iteration of the protocol in G_{h^i}, each node needs to flood in G_{h^i}. The node with the smallest id in N_{h+1^i} 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 $u \in (N^i_{h+1} - S)$ 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 which is mixed by several iterations. If it is the first iteration in G_{h^i}, each node broadcasts $|N_{h+1^i}|$ messages. For all other iterations, the whole overhead in this phase is to implement exactly a distributed DFS in G_{h^i}. So each node transmits at most d_u messages and receives at most d_u messages.

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^i} to N_{h+1^i} and edges not in the matching are directed from N_{h+1^i} to N_{h^i}. If a node receives a BFS messages 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 collects all aggregation messages from its children in T. Then it computes the optimal node v^* in $N_{h^i} \cap T$ 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 2 messages and receives at most $2d_u - 1$ messages.

4) Extending the Matching: In this phase, nodes along the path switch the corresponding matching and non-matching 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 1 message and receives at most 1 message.

The protocol runs $|N_{h+1^i}|$ iterations in G_{h^i}. In the protocol, each node in G_{h^i} transmits at most $3 \cdot |N_{h+1^i}| + d_u$ messages and receives at most $2 \cdot |N_{h+1^i}| \cdot d_u + d_u$ messages. Meanwhile, a node participates in at most two bipartite graphs. The total number of transmitted messages in the network is $\sum_u O(|N_{h+1^i}| + d_u)$ and that of received messages is $\sum_u O(|N_{h+1^i}| \cdot d_u)$. 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.

VI. Simulation Results

We evaluate the performance of our approach using simulations. Table II shows the settings. Nodes are uniformly deployed in a $100m \times 100m$ 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 (20 meters). Similar to [17], we set $Tx = 2$ and $Rx = 1$ without loss of generality. The initial energy of each node is a random number between 1 and 10. Each node generates a message in a round. We simulate our centralized approach and numerically analyze the distributed approach.

TABLE II
CONFIGURATION

Number of nodes	$100 \sim 800$
Field(meter $\times$ meter)	100×100
Sink position	(50,50)
Transmission range	20
Energy of each node	random(1,10)
Energy consumption for transmission	2
Energy consumption for reception	1
Number of runs	1000

A. Performance of the Centralized Approach

Our simulation contains two parts. In the first part, we deploy 200 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: random case scheme and worst case scheme. In the random case, 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. In the second part, we vary node density by increasing the number of nodes in the network and compare the performance of our optimal shortest path tree with the random shortest path tree. We carry out 1000 runs for each of the two parts.

1) Lifetime Performance: For the random scheme, each node randomly choose a node as its parent from its candidate parent set. After getting the tree, we compute its lifetime. In the worst case 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 the 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 our optimal scheme with these two schemes: 1000 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 5, 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 108%. In some runs, our optimal scheme can prolong the lifetime by 5 times. When compared with the worst case scheme (dashed curve in Figure 5), the lifetime ratio is even larger. Our optimal scheme has an improvement of 401% on average. The ratio is between 3 and 8 for most runs, and sometimes it even reaches 10, which is considerable. So it is necessary to apply our approach to prolong the lifetime of the network.

2) Effects of the Node Density : We deploy different number of nodes in a $100m \times 100m$ field: 200 nodes, 500 nodes and 800 nodes. Each situation is simulated for 1000 runs. We analyze their performance.

As Figure 6 shows, our optimal scheme performances better when the node density is relatively high. When 200 nodes are deployed in the field, the ratios are in the range of $1 \sim 3$ for most runs and the average value is 2.08. If the number of the nodes is 800, the ratios are concentrated in the range of $2 \sim 4$ while the average ratio is 2.78. This can be explained as follows: when the node density increases, each node has more candidate parents to choose from, so the optimal scheme will be more balanced among nodes. We can conclude that our optimal scheme is more effective when the network is dense.

B. Numerical Analysis for the Distributed Approach

In this subsection, we analyze the performance of the distributed protocol. Nodes are randomly deployed in the regular field uniformly. We design two sets of experiments and each set runs 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.

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 7 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. So 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 while the number of nodes increases from 100 to 600.

We then analyze the overhead. We use the number of messages that a sensor transmits as its overhead during the protocol. As Figure 8 shows, each node transmits 60 messages on average (dashed line) regardless of the number of the 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 $2 \sim 3$ times of the average overhead.

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

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 9 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 is 300 and reaches 202.6 for 600 nodes. The maximum convergence time (solid line) in the network also increases linearly. Both the average overhead and the maximum overhead (Figure 10) increase linearly as the network becomes denser. We explain this phenomena as follows: the number of the iterations a node participates is $|N_{h+1^i}|$ in the connected bipartite graph, and $|N_{h+1^i}|$ is closely related to the node

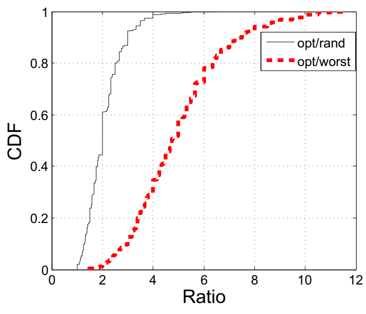

Fig. 5. Lifetime gain of the optimal scheme.

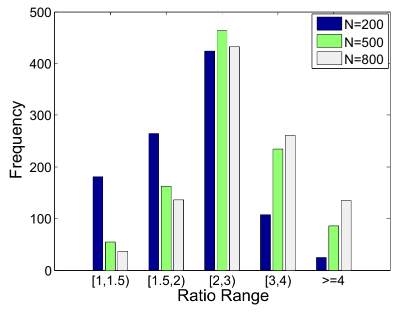

Fig. 6. Impact of the density on lifetime.

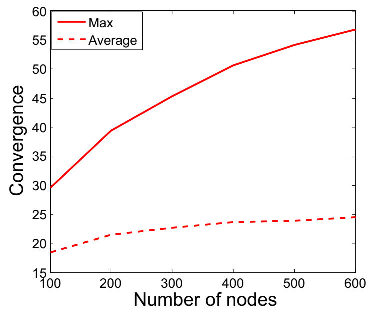

Fig. 7. Convergence time for the protocol.

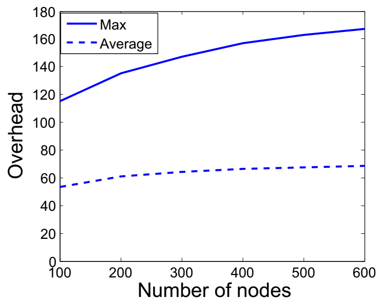

Fig. 8. Overhead for the protocol.

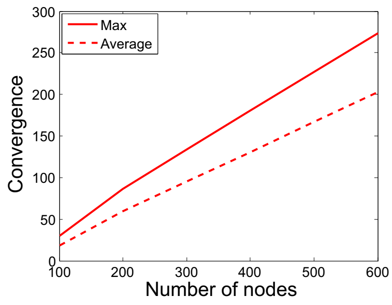

Fig. 9. Convergence time of variable densities.

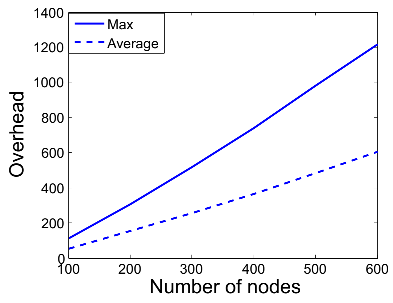

Fig. 10. Overhead of variable densities.

density. Therefore, the denser the network is, the larger the degree of each node is. So for each node u, it may have larger d_u and it is in a larger connected bipartite graph. The number of transmitted messages is about $3 \cdot |N_{h+1^i}| + d_u$. So the convergence time and the overhead increase linearly with the node density.

From above results, we conclude that the overhead of a node is determined by the density instead of the number of nodes.

VII. Conclusion and Future work

In this paper, 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 min-cost max-flow approach in polynomial time. A distributed approach has been proposed for the network. Simulation results confirm that our approach is optimal in the sense that the produced shortest path tree is optimal and outperforms greatly the random scheme.

In the future, we will study the problem of finding the maximum lifetime shortest path tree without in-network aggregation. And we will generalize the model to consider more issues such as interference, packet loss, etc.

VIII. Acknowledgement

The work is partly supported by China NSF grants (60825205, 61073152, 60903179).

References

[1] N. Xu, S. Rangwala, K. K. Chintalapudi, and D. Ganesan, "A wireless sensor network for structural monitoring," in *SenSys*, 2004.

[2] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culer, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *WSNA*, 2002.

[3] A. Giridhar and P. R. Kumar, "Computing and communicating functions over sensor networks," *IEEE Journal on Selected Areas in Communication*, 2005.

[4] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *SenSys*, 2009.

[5] C.-J. M. Liang, J. Liu, L. Luo, A. Terzis, and F. Zhao, "Racnet: a highfidelity data center sensing network," in *Sensys*, 2009.

[6] Y. Wu, S. Fahmy, and N. B. Shroff, "On the construction of a maximum-lifetime data gathering tree in sensor networks: NP-completeness and approximation algorithm," in *INFOCOM*, 2008.

[7] J. Fakcharoenphol, B. Laekhanukit, and D. Nanongkai, "Faster algorithms for semi-matching problems," in *ICALP*, 2010.

[8] N. J. Harvey, R. E. Ladner, L. Lovasz and T. Tamir, "Semi-matchings for bipartite graphs and load balancing," in *WADS*, 2003.

[9] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Maximum lifetime data gathering and aggregation in wireless sensor networks," in *ICN*, 2002.

[10] J. Park and S. Sahni, "Maximum lifetime routing in wireless sensor networks," *IEEE Transactions on Computers, 55: 1048-1056*, 2006.

[11] Y. Shi and Y. T. Hou, "Theoretical results on base station movement problem for sensor network," in *INFOCOM*, 2008.

[12] H. K. Le, D. Henriksson, and T. Abdelzaher, "A control theory approach to throughput optimization in multi-channel collection sensor networks," in *IPSN*, 2007.

[13] Y. Xue, Y. Cui, and K. Nahrstedt, "Maximizing lifetime for data aggregation in wireless sensor networks," ACM/Kluwer Mobile Networks and Applications (MONET) Special Issue on Energy Constraints and Lifetime Performance in Wireless Sensor Networks, 853-64. 2005.

[14] Y. Wu, J. A. Stankovic, T. He, and S. Lin, "Realistic and efficient multi-channel communications in wireless sensor networks," in *INFOCOM*, 2008.

[15] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in networks with power estimation," *Soviet Mathematics Doklady, 11: 1277-1280*, 1970.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 2nd ed. The MIT Press, 2004.

[17] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *MOBICOM*, 2000.