Computer Networks 57 (2013) 1063–1077

Contents lists available at SciVerse ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Network lifetime maximization for time-sensitive data gathering in wireless sensor networks

Feng Shan [a,e,*], Weifa Liang [b], Jun Luo [c], Xiaojun Shen [d]

[a] School of Computer Science and Engineering, Southeast University, Jiangsu, Nanjing 210096, China
[b] Research School of Computer Science, Australian National University, Canberra, ACT 0200, Australia
[c] School of Computer, National University of Defense Technology, Changsha, Hunan 410073, China
[d] School of Computing and Engineering, University of Missouri-Kansas City, Kansas City, MO 64110, USA
[e] Key Laboratory of Computer Network and Information Integration, Ministry of Education, Nanjing 210096, China

ARTICLE INFO

Article history:
Received 12 July 2012
Received in revised form 25 October 2012
Accepted 11 December 2012
Available online 20 December 2012

Keywords:
Wireless sensor networks
Network lifetime prolongation
Energy optimization
Load-balanced spanning tree
Network flow
Algorithm design

ABSTRACT

Energy-constrained sensor networks have been widely deployed for environmental monitoring and security surveillance purposes. Since sensors are usually powered by energy-limited batteries, in order to prolong the network lifetime, most existing research focuses on constructing a load-balanced routing tree rooted at the base station for data gathering. However, this may result in a long routing path from some sensors to the base station. Motivated by the need of some mission-critical applications that require all sensed data to be received by the base station with minimal delay, this paper aims to construct a routing tree such that the network lifetime is maximized while keeping the routing path from each sensor to the base station minimized. This paper shows that finding such a tree is NP-hard. Thus a novel heuristic called top-down algorithm is presented, which constructs the routing tree layer by layer such that each layer is optimally extended, using a network flow model. A distributed refinement algorithm is then devised that dramatically improves on the load balance for the routing tree produced by the top-down algorithm. Finally, extensive simulations are conducted. The experimental results show that the top-down algorithm with balance-refinement delivers a shortest routing tree whose network lifetime achieves around 85% of the optimum.

1. Introduction

Recent advances in electronic and communication technologies make it possible to build a large scale Wireless Sensor Network (WSN) with hundreds of thousands of sensors. Due to its wide range of applications, from environmental monitoring to mission-critical surveillance [19], WSNs have received tremendous attentions and data gathering as its fundamental function has been extensively studied in the past several years. Sensors in most WSNs are powered by energy-limited batteries, and sometimes it is impossible to recharge or replace these batteries when the network is deployed in harsh or human inaccessible environments such as battlefields or nuclear polluted regions. Therefore, energy conservation in this type of network is of paramount importance in order to prolong the network lifetime. Most existing research focused on maximizing the network lifetime by constructing a load-balanced routing tree for data gathering. However, such a tree may contain long routing paths from some sensors to the base station. In order to meet the need of mission-critical applications that require all sensed data sending their data to the base station with minimal delay, this paper aims to constructing a routing tree rooted at the base station that guarantees to forward the sensed data from

* Corresponding author at: School of Computer Science and Engineering, Southeast University, Jiangsu, Nanjing 210096, China.
E-mail addresses: shanfeng@seu.edu.cn (F. Shan), wliang@cs.anu.edu.au (W. Liang), junluo@nudt.edu.cn (J. Luo), shenx@umkc.edu (X. Shen).

1389-1286/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2012.12.005

any sensor along a shortest path while maximizing network lifetime. As we here deal with time-sensitive data gathering, we expect to collect the detailed data from all sensors without any aggregation during the data transfer. We thus assume the energy consumption of each node (sensor) is *proportional* to the number of descendants of the node in the routing tree.

This optimization problem is also applicable for non-mission-critical but large scale WSNs. This is because wireless communication (particularly multi-hop relay communication) is unreliable and long routing paths may cause frequent and repeated re-transmissions that lead to network failures. Therefore, a shorter routing path is highly desirable.

1.1. Related work

Data gathering in a WSN means collecting sensed data from every sensor and forwarding the data to the base station. The most popular paradigm of data gathering is in-network processing that constructs a routing spanning tree rooted at the base station (also referred to as a sink). Depending on different applications, two major models are used in data relaying, namely *the aggregated relay* and *non-aggregated relay* models. Most routing trees adopt the aggregated relay model. Under this model, each relay node aggregates all received data from its children and its own into a fixed-size message. The aggregated data are then transmitted to its parent. This type of data gathering is applicable to applications such as database queries AVG, MIN, MAX, COUNT, and so on. In the non-aggregated relay model, the length of a message transmitted by a relay node depends not only on the length of its own sensed data but also on the lengths of the messages received from its children. We refer to this latter one as the message-length dependent data gathering [15].

An extensively studied data gathering problem under both aggregated and non-aggregated models is to find a routing tree that minimizes the total energy consumption or minimizes the maximum energy consumption among individual nodes. Heinzelman et al. [9] initiated this study under the aggregation model and proposed the clustering protocol LEACH that groups nodes into a number of clusters in a self-organizing manner. Then, a cluster-head serves as the local 'base station' to aggregate the messages gathered from its members and forward the result to the sink directly. Lindsey and Raghavendra [17] presented an improved protocol called PEGASIS, in which all nodes in a cluster form a chain and one of them is chosen as the head responsible for reporting the aggregated result to the base station. Kalpakis et al. [13] attacked this problem by formulating it as an integer program and gave a heuristic solution.

A number of research papers have been published [2,3,7,13,15,22,23] that use various energy saving or balancing models. For example, Goel and Estrin [7] addressed the problem of minimizing the total transmission energy consumption, assuming that the aggregation cost at each relay node is a concave, non-decreasing function. They proposed a hierarchical matching algorithm that delivers an approximate solution within a logarithmic factor of the optimum. Cristescu et al. [3] studied the data correlation problem with an objective of minimizing the total transmission energy consumption. They assumed that each node is cognizant of which nodes it should be merged with so that the merged message has a minimal length. They showed that the data correlation problem is NP-complete, and provided an integer program solution, using the Slepian-Wolf coding approach. Rickenbach and Wattenhofer [21] studied the same problem and provided an improved solution with an approximation ratio of $2(1+\sqrt{2})$, using the *shallow light tree* concept [14]. Buragohain et al. [2] studied the min–max model for the network lifetime maximization problem. Instead of minimizing the total energy consumption, they focused on minimizing the maximum energy consumption among the sensors. They showed that finding an optimal routing tree under this model is NP-complete, and proposed a heuristic solution. Liang and Liu [15] also independently showed its NP-completeness and devised several heuristics that trade off between different energy optimization metrics. Intanagonwiwat et al. [12,11] studied the general data gathering issue by incorporating the semantics of an aggregation query into building an energy efficient routing tree that may not necessarily be a spanning tree. For example, they proposed a data dissemination scheme called *directed diffusion with opportunistic aggregation* [11], where data is opportunistically aggregated at relaying nodes on a low-latency tree. They also explored a greedy aggregation by a novel approach [12] that adjusts aggregation points to increase path sharing and thereby reducing the energy consumption.

With different objectives, a number of algorithms have been proposed to produce different routing (spanning) trees. For example, a Breadth-first search tree in Tiny AGgregation service (TAG) [18] aims at minimizing the transmission delay from each sensor to the root, while a degree-constrained spanning tree [2,25,24] focuses on minimizing the maximum energy consumption by any node. Another kind of spanning tree [21] makes a trade off between the energy cost of a minimum spanning tree and the energy cost of a shortest path tree. It seeks a fair balance between the total energy consumption and the maximum energy consumption among the sensors within each data gathering session. Wu et al. [25] considered the network lifetime maximization problem with the same assumption used in [2] that the size of forwarded data from each relay node is identical and the energy consumption at each node is proportional to the number of its children. They generalized the original algorithm for degree-constrained spanning trees [5] to an algorithm for routing trees in sensor networks by incorporating the residual energy into the design. Wu et al. [24] later further extended their results to a routing forest instead of a routing tree.

In addition to the above mentioned tree construction algorithms, special efforts have also been made by researchers for constructing (energy) load-balanced routing trees. For example, Hsiao et al. [10] introduced the dynamic load-balanced tree for a grid-topology of wireless access networks and developed a distributed algorithm. Dai and Han [4] introduced a *hierarchy-balanced tree* and made use of the Chebyshev sum as a measuring criterion

for top-level load-balance trees. Liang and Liu [15] considered the construction of the spanning tree dynamically with an aim to balance the transmission load among the sensors according to their residual energy so that the network lifetime can be prolonged. Yan et al. [26] extended the load-balanced tree concept by introducing the *dynamic load-balanced tree*, in which the load-balanced tree is dynamically constructed per data gathering session. However, the time and energy overhead incurred by this approach is too excessive to be acceptable for mission-critical data gathering applications. Liang et al. [16] recently considered the network lifetime maximization problem for the non-aggregation model and showed its NP-hardness. They provided an approximate solution of $\Omega(\log n/\log\log n)$, by reducing the problem to a bottleneck spanning tree problem.

However, almost all of these existing algorithms produce load-balancing or energy-saving trees without taking into account the cost of transmission delays. That is, they allow a balanced tree to be 'slim', which means the data from some leaf nodes will take a much longer journey to reach the tree root (the base station) than the average. This may cause the failure of the entire network in an unreliable wireless communication environment or lead to an intolerably long delay for mission-critical applications.

1.2. Contributions

The major contribution made by this paper is to address the need for time sensitive data gathering to guarantee minimum delay when maximizing network lifetime. Unlike previous works, this paper assumes that each sensor must send its data to the base station within the minimum number of hops (a shortest path). It appears to be the first time to formulate this type of optimization problem that has many potential applications such as disaster reliefs and military responses. The contributions to the new optimization problem by this paper can be summarized as follows.

First, it shows that finding a maximum network lifetime shortest path routing tree is equivalent to constructing a node load-balanced distance spanning tree, which is NP-hardness. Second, a novel *top-down* heuristic is presented, which makes use of the network flow technique by optimally constructing the balance spanning tree layer by layer. A distributed implementation of the proposed algorithm is also given. Third, to further improve the load-balanced spanning tree, a distributed load-balance refinement algorithm is proposed which effectively improves on the load balance for the routing tree produced by the 'top-down' algorithm. Finally, the performance of the proposed heuristics are evaluated through extensive simulations. The experimental results demonstrate that the proposed algorithms are very promising and deliver near optimal routing trees.

The rest of the paper is organized as follows. The system model and the optimization problem are defined in Section 2. The proof of NP-hardness of the problem is given in Section 3, and the top-down distance tree algorithm and the balance-refine algorithm are proposed in Section 4 and Section 5, respectively. The performance evaluation is given in Section 6. The conclusion is presented in Section 7.

2. System and problem formulation

A wireless sensor network can be modeled as an undirected, connected graph $M = (N \cup \{s\}, L)$, where N is a set of n stationary, identical sensor nodes randomly deployed in a monitoring region, s is the sink node, and L is a set of links between sensors and a sink and the sink. There is a link between sensors u and v if and only if they are within transmission range of each other. For ease of presentation, we do not distinguish a node in the graph and its corresponding sensor. We treat the sink as a special sensor that receives sensed data. We assume that every sensor has a limited initial energy *IE* while the sink has an unlimited energy supply. For the distributed implementations, we assume each node has no knowledge of the topology of the entire network but assume initially each node has its local knowledge which includes its own ID number, and its neighbors' ID numbers that can be easily obtained by a local broadcast and acknowledgment messages. For the centralized algorithm, we assume the topology is known for the computation, as other works do. It must be mentioned that in this work although we consider the single sink data gathering problem only, the developed algorithms and techniques can be easily extended to solve the similar problem in a wireless sensor network with multiple sinks. We will attempt to design an algorithm to construct a shortest routing tree rooted at s for a given network $M(N \cup \{s\}, L)$ such that the network lifetime is maximized under the non-aggregated relay model. In the rest of this paper, unless otherwise specified, we assume that a routing tree is a shortest path tree and the length of a routing path is the number of links in the path. We further assume that the transmission delay of a message from its source to the destination (the sink) is proportional to the length of its routing path. Since all the data generated from the sensors must go through the nodes adjacent to the sink, these adjacent nodes (the children of the sink in a shortest routing tree) will consume much more energy than others. Balancing the load among them is the key to prolonging the network lifetime. The optimal balancing is obtained if the number of nodes in the maximum branch (subtree) rooted at a child node of s, denoted by *NMB*, is minimized. In order to formally define the optimization problem dealt with by this paper, we introduce the following notations and assumptions.

(a) For simplicity, this paper only takes into account the energy consumption by each sensor for transmission and reception. This is because the radio frequency transmission is the dominant energy consumption in wireless networks [20]. Let e_t and e_r (usually $e_t > e_r$) denote the amounts of energy consumed by a sensor node for transmitting and receiving one bit of data, respectively.

(b) For a given data gathering session, we assume that the size of sensed data by any sensor is identical to a fixed length l.

(c) Given a routing tree T rooted at the sink, for each node v in T, let $nd(v)$ denote the number of nodes in the subtree rooted at v including node v itself. The amount of energy consumed by node v per data gathering session $e_c(v)$ can be calculated by

$$e_c(v) = (nd(v) - 1) \cdot l \cdot e_r + nd(v) \cdot l \cdot e_t = l \cdot (nd(v) \cdot (e_r + e_t) - e_r).$$

Obviously, a node in each data gathering session consumes more energy than any of its descendants.

(d) Let $Adj(s) = \{v_1, v_2, \ldots, v_d\}$ denote the set of neighboring nodes of sink s in $M(N \cup \{s\}, L)$. Obviously, these nodes must also be the children of s in any shortest path tree.

(e) Suppose T is a routing tree. We define $NMB(T) = \max_{v \in Adj(s)}\{nd(v)\}$, where *NMB* stands for the Num-ber of nodes in the Maximum Branch.

(f) Suppose T is a routing tree. Let node $v_k \in Adj(s)$ be such a child of s, so that $nd(v_k) = \max_{v \in Adj(s)}\{nd(v)\} = NMB(T)$. Then, $e_c(v_k) = \max_{v \in N} e_c(v)\}$. Therefore, the lifetime of $M(N \cup \{s\}, L)$ for data gathering can be defined by

$$Life(N, L, T) = \frac{IE}{e_c(v_k)}, \quad (1)$$

where $v_k \in Adj(s)$ with $nd(v_k) = NMB(T)$.

Definition 1. Given a sensor network $M(N \cup \{s\}, L)$, *the problem of maximum lifetime shortest routing tree for data gathering* is to construct a shortest path routing tree T rooted at s such that $Life(N, L, T)$ is maximized, subject to meeting the assumptions from (a) to (f). We refer to tree T as the *maximum lifetime shortest routing tree*.

Clearly, this problem is equivalent to finding a routing tree T such that *NMB(T)* is minimized.

3. NP-Hardness

In this section we show that, given a graph $M = (N \cup \{s\}, L)$, finding the maximum lifetime shortest routing tree is NP-hard.

Definition 2. An h-layer graph is a graph *G(V, E)*, where the node set V is partitioned into disjoint subsets,

$V_0, V_1, \ldots, V_h$, and the nodes in V_i are in layer i, $0 \leqslant i \leqslant h$. Moreover, $V_0 = \{s\}$ consists of a single node called the source (or root). A node in layer $i, 1 \leqslant i \leqslant h$, may be only adjacent to nodes in layer $i - 1$ or layer $i + 1$. There are no edges within each layer or connecting nodes of non-adjacent layers. An h-layer graph, $h \geqslant 1$, is called a multi-layer graph.

Let $dist(a, b)$ denote the distance from node a to node b in a graph that is the number of edges used by a shortest path from a to b. Obviously, in an h-layer graph, V_i consists of all nodes whose distance to s is $i, 1 \leqslant i \leqslant h$.

Definition 3. Given a connected graph *G(V, E)* and a node $s \in V$, its distance graph $G' = (V', E')$ rooted at s is a subgraph of G such that $V' = V$ and $E' = \{(u, v) | (u, v) \in E$ and $dist(u, s) + 1 = dist(v, s)\}$.

Obviously, the distance graph $G(V', E')$ is a multi-layer graph that can be easily obtained by performing the Breadth-first search starting from node s.

Definition 4. Given a connected graph *G(V, E)* and a node $s \in V$, a spanning tree T rooted at s is called a distance spanning tree if the length of any path in T from $v \in V$ to s is equal to $dist(v, s)$.

Definition 5. Given a connected graph *G(V, E)* and a node $s \in V$, a distance spanning tree T rooted at s is called node balanced if the value of *NMB(T)* is minimized.

Obviously, any distance spanning tree in *G(V, E)* rooted at s is a subgraph of the distance graph $G(V', E')$ of G. It is also clear that, given a sensor network $M(N \cup \{s\}, L)$, any shortest path routing tree is a distance spanning tree rooted at s and vice versa. Therefore, following the discussion in Section 2, finding the maximum lifetime shortest routing tree in $M(N \cup \{s\}, L)$ is equivalent to finding a node load-balanced distance spanning tree in its distance graph. Let us denote this problem by *NBDT*, namely, *the Node Balanced Distance Tree*.

Theorem 1. *Given a connected graph G(V, E) and a node $s \in V$, the NBDT problem is NP-hard even for the special case where $|Adj(s)| = 2$.*

Proof. We reduce the set cover problem [6] to the *NBDT* problem. An instance of the set cover problem can be stated as follows. Given a positive integer k ($\leqslant m$) and a family $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$ of m sets whose union consists of n different elements, $e_1, e_2, \ldots, e_n$, is there a collection $\mathcal{C}$ of k sets selected from the m sets such that $\bigcup_{S_i \in \mathcal{C}} S_i = \{e_1, e_2, \ldots, e_n\}$?

Let $U = \bigcup_{S_i \in \mathcal{C}} S_i = \{e_1, e_2, \ldots, e_n\}$. We now transform this instance to an instance of the *NBDT* problem in a 4-layer graph G whose construction is illustrated in Fig. 1 as follows.

(1) The layer $V_1 = \{u_1, u_2\}$ contains two nodes and both are adjacent to s so that $|Adj(s)| = 2$.

(2) For layer V_2, we create a node S_i for every set S_i in $\mathcal{F}$ and connect each of them to both nodes u_1 and u_2 by edges (u_1, S_i) and (u_2, S_i), $(1 \leqslant i \leqslant m)$. Moreover, we create $mn + 2k - m$ additional nodes for layer V_2, and connect each of them to u_2 only.

(3) For layer V_3, we create n nodes, $e_1, e_2, \ldots, e_n$, that correspond to the n different elements in U. Node e_j is connected to node S_i in V_2 by an edge (S_i, e_j) if and only if $e_j \in S_i, 1 \leqslant j \leqslant n$ and $1 \leqslant i \leqslant m$.

(4) For layer V_4, we create $m - 1$ nodes for each node $e_j \in V_3$ and connect each of them to node e_j by an edge, $1 \leqslant j \leqslant n$.

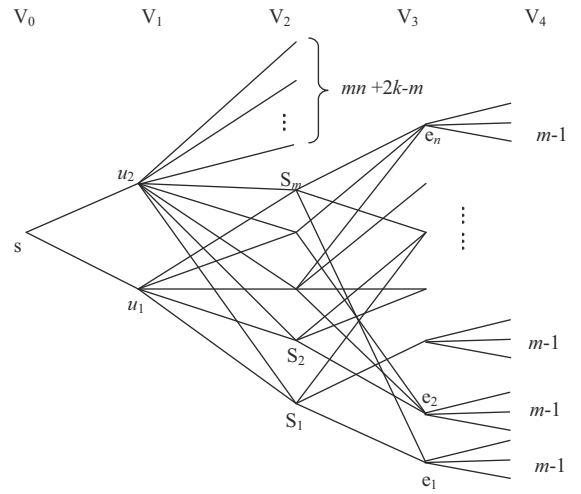


Fig. 1. An illustration of the 4-layer graph G transformed from an instance of the set cover problem.

The transformation takes polynomial time of n and m. The number of nodes in G is calculated as follows.

There are two nodes in V_1, $mn+2k$ nodes in V_2, n nodes in V_3, and $n(m-1)$ nodes in V_4. Therefore, the total number of nodes in graph G is $2+mn+2k+n+n(m-1)=2(mn+k+1)$. Thus, any distance spanning tree T must have $NMB(T) \geqslant mn+k+1$. In the following we show that there is a solution to the instance of the set cover problem if and only if there is a distance spanning tree T in G such that $NMB(T)=mn+k+1$.

We show the *only if* part first. Suppose there is a solution to the instance of the set cover problem and $\mathcal{C}$ is the collection of k sets such that $\bigcup_{S_i \in \mathcal{C}} S_i = \{e_1, e_2, \ldots, e_n\}$. We construct a distance spanning tree T as follows.

(1) Connect s to u_1 and u_2.
(2) Connect u_1 to $S_i \in V_2$ by edge (u_1, S_i) if $S_i \in \mathcal{C}$.
(3) Connect u_2 to each remaining node in V_2. Because there are $m-k$ sets that are not in $\mathcal{C}$, u_2 has $(m-k)+mn+2k-m+1=mn+k+1$ children, including u_2 itself.
(4) For element $e_j \in U$, find a set $S_i \in \mathcal{C}$ such that $e_j \in S_i$. Then, connect corresponding node $S_i \in V_2$ to node $e_j \in V_3$. Because the collection $\mathcal{C}$ covers all elements, this can be done. Obviously, all nodes in V_3 become the descendants of u_1.
(5) Include all edges between V_3 and V_4. All nodes in V_4 become the descendants of u_1 in the tree.

Following the above construction, it is clear that each of the nodes u_1 and u_2 has exactly $mn+k+1$ descendants. Therefore, $NMB(T)=mn+k+1$.

We then show the *if* part. Suppose there is a distance spanning tree T such that $NMB(T)=mn+k+1$. Then, the subtree rooted at u_1 must connect to at least k nodes in V_2; otherwise, it would have less than $mn+k+1$ descendants even if all nodes in layers V_3 and V_4 are its descendants. However, u_1 cannot connect to more than k nodes in V_2 either. Suppose, for the sake of contradiction, u_1 connects to more than k nodes in V_2. To be balanced, the tree rooted at u_2 must contain at least one node x in layer V_3, while node x must connect the $(m-1)$ nodes in V_4 that are exclusively adjacent to x only. Then, the tree rooted at u_2 must have at least $1+(mn+2k-m)+1+1+(m-1)=(mn+k+1)+(k+1)$ nodes, contradicting that the tree is balanced. Therefore, in the balanced tree, node u_1 must connect to exactly k nodes in V_2. Moreover, the k nodes must connect to all n nodes in V_3 and then all nodes in V_4, in order for u_1 to have $(mn+k+1)$ nodes in its subtree. Obviously, these k nodes in V_2 correspond to the k sets that cover all n elements in U. The theorem then follows. □

4. Top-down distance tree algorithm

As discussed in Section 2, finding the maximum lifetime shortest routing tree in $M(N \cup \{s\}, L)$ is equivalent to finding a node load-balanced distance spanning tree in its distance graph. For ease of presentation, in this section, the notation $M(N \cup \{s\}, L)$ is also used to represent its distance graph if no ambiguity arises. Denote by V_l the set of nodes in layer l and E_l the set of edges between layer l and layer $l+1$ in $M(N \cup \{s\}, L)$. Let T_l be a distance spanning tree up to layer l and $nd_l(v_k)$ the number of descendants of node v_k in $T_l, 1 \leqslant l \leqslant h$. Equivalently, T_l can be viewed as the subtree of a distance spanning tree that consists of all nodes from layer 0 up to layer l.

4.1. A centralized algorithm

Given a distance spanning tree $T_l, 1 \leqslant l \leqslant h-1$, we can extend it to layer $l+1$ by linking each node $v \in V_{l+1}$ to a node $u \in V_l$ through an edge $(u, v) \in E_l$. Starting from T_1, the top-down heuristic uses the network flow technique to repeatedly and optimally extend tree T_l to $T_{l+1}, 1 \leqslant l \leqslant h-1$, until all the nodes in N are included in the tree.

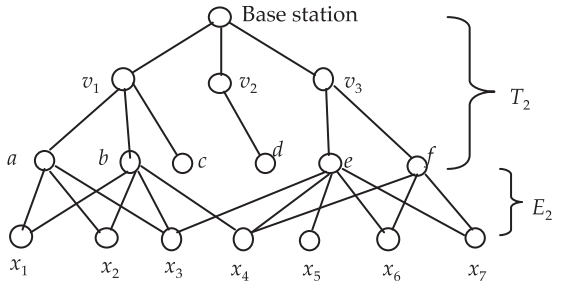
Definition 6. A distance spanning tree T_{l+1} is said to be optimally extended from $T_l, 1 \leqslant l \leqslant h-1$, if $NMB(T_{l+1})(=\max_{1 \leqslant k \leqslant d}\{nd_{l+1}(v_k)\})$ is minimized among all possible extensions from layer l to layer $l+1$, where $nd_{l+1}(v_k)$ denotes the number of nodes in the subtree of T_{l+1} rooted at v_k, $v_k \in Adj(s), k=1,2,\ldots,d$.

In the following we explain how this heuristic optimally extends tree T_l to $T_{l+1}, 1 \leqslant l \leqslant h-1$. Because the network flow technique is the key technique employed in this algorithm, we first describe how to construct the flow network according to tree $T_l, 1 \leqslant l \leqslant h-1$.

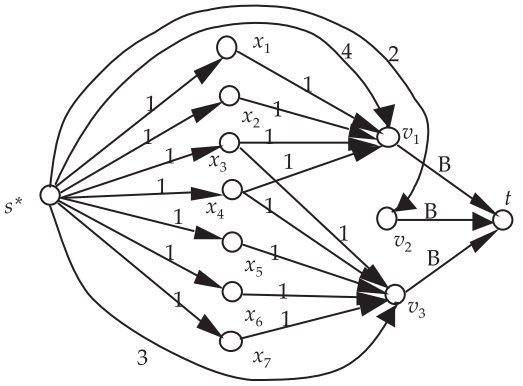
4.1.1. Constructing a flow network N(B) from the given T_l

Let $A(v)=v_k$ denote that $v \in N$ is a descendant of $v_k \in Adj(s)$ in T_l, and let $V_{l+1}=\{x_1, x_2, \ldots, x_m\}$ be the set of nodes in layer $l+1$. The construction of the flow network *N(B)* is given below.

As part of *N(B)*, construct a directed bipartite graph *G (X, Y, E)*, where $X=V_{l+1}=\{x_1, x_2, \ldots, x_m\}, Y=Adj(s)=$



(a) The distance spanning tree T_2 and the set E_2 of edges between layers 2 and 3.



(b) Graph $N(B)$ constructed for extending the distance spanning tree T_2 to T_3.

Fig. 2. An illustration of the construction of flow network $N(B)$.

$\{v_1, v_2, \ldots, v_d\}$, and $E = \{(x, v) \mid x \in X, v \in Y, \exists u$ such that $(x, u) \in E_l$ and $A(u) = v\}$, where node $v \in Adj(s)$ is an ancestor of node $u \in V_l$ in T_l. The meaning of edge (x, v) is that x can become a descendant of v through edge (x, u). There can be multiple edges from x to different nodes in set Y. The capacity of every edge in $G(X, Y, E)$ is set 1. Add to $N(B)$ a source node s^* and a directed edge (s^*, x) for every node $x \in X$, with capacity $c(s^*, x) = 1$. Add to $N(B)$ a directed edge (s^*, v) for every node $v \in Y$, with capacity $c(s^*, v_k) = nd_l(v_k)$. Add to $N(B)$ a sink node t and a directed edge (v, t) for every node $v \in Y$, with a capacity $c(v, t) = B$, where B is an adjustable integer whose meaning will be clear later. Fig. 2 shows an example of the construction of $N(B)$.

Lemma 1. *Given a positive integer B as the edge capacity in $N(B)$ with $1 \leqslant B \leqslant |N|$, there exists a maximum integral flow f in $N(B)$ that saturates all outgoing edges from s^* if and only if there is a distance spanning tree T_{l+1} extended from T_l such that $\max_{1 \leqslant k \leqslant d}\{nd_{l+1}(v_k)\} \leqslant B$.*

Proof. We first show the *only if* part. Let f be a maximum flow in $N(B)$ that saturates all outgoing edges from s^*. We can extend T_l in the following way. Because $f(s^*, x_i) = 1$, every node x_i must have exactly one outgoing edge (x_i, v_k) with flow $f(x_i, v_k) = 1, 1 \leqslant k \leqslant d, 1 \leqslant i \leqslant m$. Since $(x_i, v_k) \in E$, by the construction of $N(B)$, there is a node $u \in V_l$ such that $(x_i, u) \in E_l$ and $A(u) = v_k$. We thus connect x_i to node u in T_{l+1}. Obviously, x_i becomes a descendant of v_k through edge (x_i, u). By doing so, we connect every $x_i \in V_{l+1}, 1 \leqslant i \leqslant m$, to a node in V_l, which effectively extends T_l to T_{l+1}. Moreover, for each node $v_k, 1 \leqslant k \leqslant d$, we have $f(v_k, t) = \sum_{i=1}^{m} f(x_i, v_k) + f(s^*, v_k)$. Because $\sum_{i=1}^{m} f(x_i, v_k)$ is equal to the number of nodes in layer $l+1$ that become new descendants of v_k and $f(s^*, v_k) = nd_l(v_k)$ is the number of descendants of v_k from layer 1 to layer l, we have $f(v_k, t) = nd_{l+1}(v_k)$ in the extended T_{l+1}. Therefore, $nd_{l+1}(v_k) = f(v_k, t) \leqslant B$.

We then prove the *if* part. Suppose T_{l+1} is a distance spanning tree extended from T_l such that $\max_{1 \leqslant k \leqslant d}\{nd_{l+1}(v_k)\} \leqslant B$. According to T_{l+1}, we assign a flow f in $N(B)$ as follows: assign $f(s^*, x_i) \leftarrow 1$ and $f(x_i, v_k) \leftarrow 1$ if x_i is connected to a node $u \in V_l$ in T_{l+1} and $A(u) = v_k$; assign $f(s^*, v_k) \leftarrow nd_l(v_k)$ and $f(v_k, t) \leftarrow nd_{l+1}(v_k)$; and assign all other edges with zero flow, $1 \leqslant i \leqslant m$ and $1 \leqslant k \leqslant d$. Obviously, this flow assignment saturates all outgoing edges from s^*, and because $\max_{1 \leqslant k \leqslant d}\{nd_{l+1}(v_k)\} \leqslant B$, the amount of flow assigned on any edge is no greater than its capacity. Moreover, only if x_i is a descendant of v_k in T_{l+1}, then we assign $f(x_i, v_k) = 1$; otherwise $f(x_i, v_k) = 0$. So, $\sum_{i=1}^{m} f(x_i, v_k)$ is equal to the number of newly added descendants from layer $l+1$ to v_k. Therefore, we have

$$f(v_k, t) = nd_{l+1}(v_k) = \sum_{i=1}^{m} f(x_i, v_k) + nd_l(v_k) = \sum_{i=1}^{m} f(x_i, v_k) + f(s^*, v_k). \quad (2)$$

Thus, at any node except s^* and t, the total incoming flow is equal to the total outgoing flow. Therefore, the assigned flow is a valid flow and it saturates all outgoing edges from s^*. Lemma 1 then follows. □

Corollary 1. *Finding an optimally extended distance spanning tree T_{l+1} from tree T_l is equivalent to finding a minimum integer B in the flow network $N(B)$ such that a maximum flow can saturate all outgoing edges from $s^*, 1 \leqslant B \leqslant |N|$.*

Proof. This corollary follows Lemma 1, omitted. □

We now determine the minimum integer B in the flow network $N(B)$ such that a maximum flow can saturate all outgoing edges from s^*. Following the construction of $N(B)$, to saturate all outgoing edges from s^* in $N(B)$, there is such an integer B that satisfies the following inequality for a flow f:

$$\max_{1 \leqslant k \leqslant d}\{nd_l(v_k)\} \leqslant B \leqslant \max_{1 \leqslant k \leqslant d}\{nd_l(v_k)\} + m,$$

where $m = |V_{l+1}|$. The smallest value of B can be found by algorithm `Smallest_B` using binary search as follows.

Algorithm 1. `Smallest_B`$(N(B), T_l, m)$

1: $lower\_bound \leftarrow \max_{1\leqslant k\leqslant d}\{nd_l(v_k)\}$;
/* This was known when T_l was produced */
2: $upper\_bound \leftarrow lower\_bound + m$;
3: $c \leftarrow \sum_{i=1}^{d} nd_l(v_k)$;/* The total capacity on all of $(s^*, v_k), 1 \leqslant k \leqslant d$ */
4: **while** $lower\_bound \neq upper\_bound$ **do** 5:
$b \leftarrow \lfloor \frac{upper\_bound+lower\_bound}{2} \rfloor$;
6: $B \leftarrow b$ /* Set $B = b$ in $N(B)$ */;
7: Find a max flow f in $N(B)$ from s^* to t;
8: **if** $|f| = m + c$ **then**
9: /* f saturates all out edges from s^* */
10: $upper\_bound \leftarrow b$;
11: **else**
12: $lower\_bound \leftarrow b + 1$;
13: **end if**
14: **end while**
15: $B \leftarrow upper\_bound$;
16: **return** B.

Theorem 2. *Algorithm* **Smallest-***B correctly finds the smallest integer B for the flow network N(B) such that the maximum flow saturates all outgoing edges from* s^*.

Proof. It is clear that the value of the smallest B in $N(B)$ is between the initial lower bound and the upper bound, namely, $\max_{1\leqslant k\leqslant d}\{nd_l(v_k)\} \leqslant B \leqslant m + \max_{1\leqslant k\leqslant d}\{nd_l(v_k)\}$. The *while* loop in the `Smallest_B` algorithm reduces the searching space by at least a half after each iteration, but guarantees that the smallest B is still within the reduced interval. Therefore, when the length of the interval becomes 0, the smallest B is found. □

4.1.2. The top-down distance tree algorithm

The following top-down distance tree heuristic algorithm constructs a distance spanning tree by repeating optimal layer extensions until all the nodes in N are included in the tree.

Algorithm 2. `Top-down-distance-tree`$(M(N \cup \{s\}, L))$

1: Construct the tree T_1 by including all edges of $(s, v_k), 1 \leqslant k \leqslant d$;
2: **for** $l \leftarrow 1$ to $h - 1$ **do**
3: Construct the network graph $N(B)$ having T_l, E_l and V_{l+1};
4: $m \leftarrow |V_{l+1}|$;
5: `Smallest_B` $(N(B), T_l, m)$;
6: Find a maximum flow f in $N(B)$ from s^* to t;
7: Convert the flow f to V_{l+1} according to steps given by Lemma 1.
8: **end for**

The correctness of algorithm `Top-down-distance-tree` is obvious. In the rest of this paper, we refer to this algorithm as the 'top-down' algorithm for short.

4.2. Overview of the distributed implementation

We assume that the sensor network is a synchronous network, in which each node starts a message transmission in the beginning of a time unit, and finishes the transmission at the end of the time unit. All local computation can be done in the same time unit as well. In other words, we assume that local computation takes no time.

The distributed implementation of the 'top-down' algorithm consists of $h - 1$ iterations. Within each iteration, it extends the current tree one layer further. We now consider one layer expansion by considering a subgraph $G_{l,l+1} = (V_l \cup V_{l+1}, (V_l \times V_{l+1}) \cap L)$. Note that $G_{l,l+1}$ may not be connected. The flow network $N'(B) = (V_l \cup V_{l+1} \cup V_1, E')$ based on $G_{l,l+1}$ is constructed as follows. There is a source s^* and a destination t in $N'(B)$ such that s^* connects to all nodes in V_{l+1} while all nodes $v_i \in V_1$ $(= Adj(s))$ connect to node t, there is a directed edge from a node $y \in V_l$ to a node $v_i \in V_1$ if y is a descendant of v_i in T_l and the capacity of the edge is assigned as $B - nd_l(v_i)$. Meanwhile, the capacity of each edge from a node $v \in V_1$ to t is assigned an integer $B - nd_l(v)$ and the capacity of each edge from s^* to $x \in V_{l+1}$ is assigned an integer 1. Each edge (u,v) from a node $u \in V_{l+1}$ to a node $v \in V_l$ in $(V_l \times V_{l+1}) \cap L$ is assigned capacity of 1. The task is to find a maximum flow from s^* to t distributively to saturate all edges starting from s^*. Clearly, the flow network $N'(B)$ is equivalent to $N(B)$ defined in the previous section. We first embed the graph $N'(B)$ into the communication network $M(N \cup \{s\}, L)$ as follows. The subgraph $G_{l,l+1}$ can be embedded into the original sensor network M easily. We embed each edge from a node $y \in V_l$ to a node $v \in V_1$ into node y and each edge from $v \in V_1$ to the virtual node t into node v. Similarly we embed each edge from the virtual node s^* to a node $x \in V_{l+1}$ into node x, and embed the virtual node t into the base station.

To simulate an edge in $N'(B)$ from a node $y \in V_l$ to $v \in V_1$ (or t), or from s^* to a node $x \in V_{l+1}$ in the real communication topology M, we use the unique path in the partial BFS tree T_l^{BFS} or T_{l+1}^{BFS} for such a propose. Thus, although $G_{l,l+1}$ may not be connected, the messages sent by the nodes in it can be collected, using trees T_l^{BFS} or T_{l+1}^{BFS}. In other words, each message transfer between two neighboring nodes in $N'(B)$ can be emulated at most in $O(l)$ time with $O(l)$ messages along the unique path in the partial tree T_l^{BFS} or T_{l+1}^{BFS}. The distributed implementation is that each child $v \in Adj(s)$ of the root s broadcasts its identity and its number of descendants $nd_l(v)$ to its descendants in V_l using T_l^{BFS}. Thus, each node in V_l is labeled with one of the children of node s. Consequently, if finding a maximum flow in network $N'(B)$ from s^* to t takes $O(t_{l,l+1})$ time and $O(m_{l,l+1})$ messages, it takes $O(l \cdot t_{l,l+1})$ time and $O(l \cdot m_{l,l+1})$ messages in the original communication network $M(N \cup \{s\}, L)$.

4.3. Distributed implementation

We here give a distributed implementation of the proposed *'top-down'* algorithm, and we state the result by the following theorem.

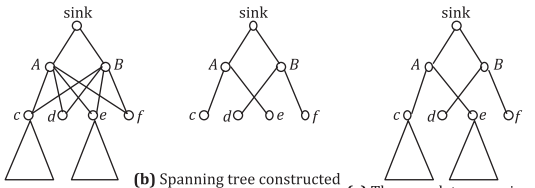
Theorem 3. *Given a wireless sensor network $M = (N \cup \{s\}, L)$, there is a distributed implementation of the 'top-down' algorithm for finding a maximum lifetime shortest routing tree, which takes $O(h \cdot |N|^2 \cdot \log |N|)$ time and uses $O(|N|^2 \cdot |L|)$ messages.*

Proof. Given the communication network $M(N \cup \{s\}, L)$, the distributed construction of a BFS tree in M rooted at the base station takes $O(h)$ time and $O(|L| + |N|^{1.6})$ messages by a distributed algorithm in [1], where h is the depth of the BFS tree. The construction of flow network $N'(B)$ and its embedding to the communication network M takes $O(|N|)$ time and $O(|L|)$ messages because the degree of each node is no more than $|N|$. Finding a maximum flow from s^* to t in $N'(B)$ takes $O(t_{l,l+1}) = O((|V_l| + |V_{l+1}|)^2)$ time and uses $O(m_{l,l+1}) = O((|V_l| + |V_{l+1}|)^2 \cdot |(V_l \times V_{l+1}) \cap L|)$ messages by Goldberg and Tarjan's distributed algorithm [8], while it takes $O(n'^2)$ time and uses $O(n'^2 m')$ messages in a graph with n' nodes and m' edges (see Th. 6.3 in [8]). Following algorithm `Smallest_B`, there are at most $\lceil \log B \rceil$ calling of the s-t maximum flow algorithm in order to find a load-balanced matching for all nodes in V_{l+1} and $1 \leqslant B \leqslant |N|$. Thus, each layer extension of the routing tree in the original sensor network M takes $O(l \cdot \log |N| \cdot t_{l,l+1})$ time and uses $O(l \cdot \log |N| \cdot m_{l,l+1})$ messages.

The algorithm for finding a maximum lifetime shortest routing tree needs $h - 1$ iterations, the total time for the routing tree construction thus is $\sum_{l=1}^{h-1} O(l \cdot \log |N| \cdot t_{l,l+1}) = O(h \cdot |N|^2 \cdot \log |N|)$ and the number of messages used is $\sum_{l=1}^{h-1} O(l \cdot \log |N| \cdot m_{l,l+1}) = O(|N|^2 \cdot |L| \cdot \log |N|)$. The theorem then follows. □

5. The balance-refine algorithm

Although the 'top-down' algorithm constructs a load-balanced spanning tree optimally layer by layer, the balance load (*NMB*) of the tree could be further improved. Fig. 3 shows such an example, where the sink has two children A and B in layer 1, and both A and B connect to 4 nodes, c, d, e, f in layer 2. So, the 'top-down' algorithm extends the tree to layer 2 by assigning nodes c and e to A and nodes d and f to B. It is a perfectly balanced tree up to layer 2. However, if node c and e can reach many nodes below layer 2, but nodes d and f connect to no nodes below layer 2 at all, then the result will be an unbalanced tree shown in Fig. 3c. This is because the 'top-down' algorithm has no way to use the connection information below the layer it is dealing with within each iteration.



(a) An original graph **(b)** Spanning tree constructed by the top-down algorithm up to layer 2 **(c)** The complete spanning tree by the top-down algorithm

Fig. 3. An example of the spanning tree constructed by the 'top-down' algorithm that needs an improvement.

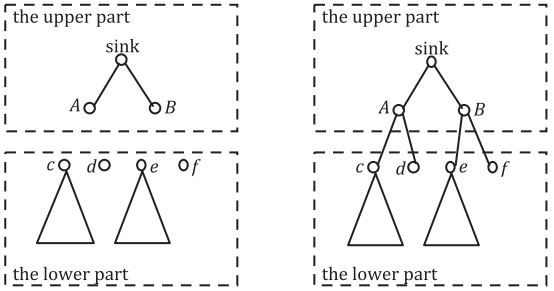
In this section, we introduce a distributed *balance-refine algorithm* to refine the tree produced by the 'top-down' algorithm, layer by layer, through changing the connection between two adjacent layers such that the load balance among the children of the root can be further improved. Specifically, for layer l and layer $l+1$ $(1 \leqslant l \leqslant h-1)$, we first remove all the edges between these two layers in the current tree so that the tree is partitioned into two parts, the upper part is the tree containing all nodes up to layer l, and the lower part consists of a set of subtrees whose roots are the nodes in layer $l+1$. As an example, Fig. 4a shows the two parts after removing all tree edges between layers 1 and 2 of the spanning tree of Fig. 3c. After removing these edges, we re-connect the two parts into a new tree using all available edges in the set $E_l = (V_l \times V_{l+1}) \cap L$ that includes previously non-tree edges as well as tree edges, such that the new tree has a better load balance. Fig. 4b shows a possible result by re-connecting the two parts of Fig. 4a which has a better load balance.

Note that the removing and re-connecting procedure on two adjacent layers is essentially different from expanding the current tree to include the nodes in one more layer. Because the lower part includes the subtrees rooted at the nodes in the next layer, and different subtrees contain different numbers of nodes, finding an optimal connection such that the new tree has the smallest *NMB* number becomes difficult and can be easily shown to be NP-complete.

5.1. Modeling of the re-connecting problem

Recall that $V_1 = Adj(s) = \{v_1, v_2, \ldots, v_d\}$. We aim to re-connect layer l and layer $l+1$ $(1 \leqslant l \leqslant h-1)$ of the current tree such that the $NMB(T) = \max_{v_i \in V_1}\{nd(v_i)\}$ in the new tree T is improved, where $nd(v_i)$ is the number of nodes in the branch rooted at v_i. To help readers understand the following distributed algorithm, we use a daily life example – the school admission process as an illustration of the proposed distributed algorithm.

We view the subtree rooted at v_i as a college i $(1 \leqslant i \leqslant d)$ and the nodes in this subtree as the students admitted to this college. The node v_i plays the role of admission officer. A node y in layer l is considered to be a recruiter for college i if y belongs the subtree rooted at v_i. Since we have removed the edge connections between layer l and layer $l+1$, the tree in the upper part represents



(b) The two parts after breaking edges **(a)** The new spanning tree after re-connecting the two parts

Fig. 4. An illustration of reconnecting layers 1 and 2 of the tree in Fig. 3c.

the current enrollments to the d colleges. Connecting a subtree rooted at node x in layer $l+1$ to a node y in layer l is equivalent to admitting all students in the subtree of x to the college where the node y belongs to. Our objective is to make the enrollments among d colleges to be balanced as much as possible. The decision on whether to admit students of a subtree is not made by the recruiter but by the admission officer only. This is because there may be many recruiters for the same college and they have no way to communicate directly with each other. We can expect that a recruiter passes an application from a subtree to its branch root (admission officer) and gets a reply back in exactly $2l$ time units, assuming that it takes one unit time for a node to send a message to a neighboring node. Fig. 5 illustrates this model by an example. Note that a college does not admit students individually but admits students in groups. The links (edges) between layer l and layer $l+1$ define all possible ways that student groups can apply for colleges. It is very likely that a student group (a subtree) may have links to multiple recruiters of the same college. For example, in Fig. 5, u_3 has links to B_2 and B_3, two recruiters for college B. In this case, the student group chooses exactly one recruiter to communicate with.

5.2. A distributed algorithm for re-connecting two layers

In this subsection, we present a distributed algorithm for re-connecting layer l and layer $l+1$ $(1 \leqslant l \leqslant h-1)$. For the sake of convenience, we make the following reasonable assumptions.

(1) Each college admission officer knows the number of students that have been recruited by the college in the upper part. A variable $enrollment(i)$ is used to denote the enrolled number of students in college i, which is updated immediately whenever the college has admitted new students.
(2) There are $k(=|V_{l+1}|)$ subtrees in the lower part whose roots are $u_1, u_2, \ldots, u_k$. We also assume that subtree rooted at u_j has s_j students, $1 \leqslant j \leqslant k$, and node u_j knows the value s_j.

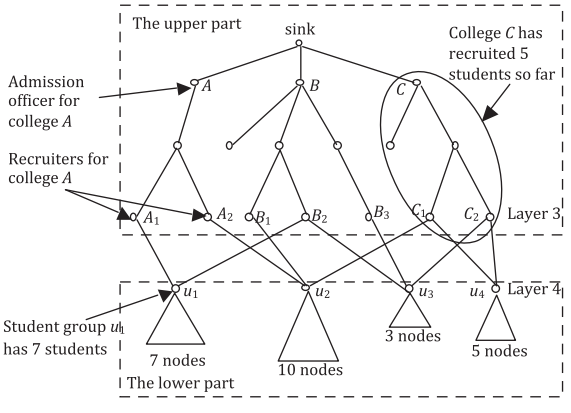


Fig. 5. An illustration of the college admission modeling.

(3) The communication between a recruiter of a college and its admission officer is through a message which is forwarded by the unique path in the tree of the upper part between them.
(4) In case a student group has links to multiple recruiters of the same college, one of the recruiters is chosen to pass the message between the student group and the admission officer of that college, and the other recruiters do not communicate with the student group.

Having the above assumptions, it can be seen that there are exactly l time units used for one-way transmission between a student group and a college admission officer, and $2l$ time units for a round trip communication. Now, we are ready to introduce the re-connection algorithm consisting of the following three stages.

5.2.1. The first stage – start stage

The re-connection procedure starts when the sink sends a message $start(nmb,$ layer $l)$ to nodes in $V_1 = \{v_1, v_2, \ldots, v_d\}$, where nmb is the NMB value of the current spanning tree which is to be improved. Upon receiving the message, each node v_i, the admission officer for college i, broadcasts a message $recruit(nmb,$ layer $l, enrollment(i),$ college $i)$ to its recruiters, $1 \leqslant i \leqslant d$. Then, each recruiter of college i broadcasts this message to all student groups the recruiter is responsible to communicate with. After receiving this message, each node enters the second stage.

5.2.2. The second stage – admission stage

In this stage, each student group applies to a college and the college selects only one student group to admit, and this interaction will repeat in every $2l$ time units until all student groups are admitted or one student group sends an *abort* message. The detailed description of this stage is as follows.

First, each student group u_j takes the following actions, $1 \leqslant j \leqslant k = |V_{l+1}|$.

(1) Upon receiving all $recruit(nmb,$ layer $l, enrollment(i),$ college $i)$ messages, it updates its local variable $enrollment(i)$. If u_j has not been admitted by any college, then it does the following:
  (a) Compute $\Delta_{ij} = enrollment(i) + s_j$. A college i is *qualified* to be considered by u_j if $\Delta_{ij} \leqslant nmb$;
  (b) If no college is qualified, then send an *abort* message to a recruiter of college i and stop communication until the third stage;
  (c) If one or more colleges is qualified, find the one with the minimum Δ_{ij}, and send a message $apply(\text{college}i, \text{group}j, \Delta_{ij}, urgent)$ to the recruiter of college i, where *urgent* is a boolean variable which will be 'true' if only college i is qualified;
(2) Upon receiving a *termination* message, it enters the third stage;
(3) Upon receiving an *abort* message, it resets the parent of u_i to be the initial parent, and then enters the third stage;
(4) Upon receiving an *admitted* message from college i, it does the following:

(a) Send a message *done* to all recruiters that node u_j communicates with;
(b) Connect u_j to the recruiter of college i. The recruiter node becomes the parent of u_j in the new tree.

Second, the main role of each recruiter in layer l is to pass messages between the college admission officer and students groups. Additional actions taken by a recruiter for college $i, 1 \leqslant j \leqslant d$, are:

(1) Upon receiving the *done* message from a student group u_j, it stops communication with u_j until the third stage;
(2) Upon receiving *done* messages from all student groups with which the recruiter is responsible for communication, it sends a *done* message to college admission officer v_i.

Third, each college admission officer $v_i, 1 \leqslant i \leqslant d$, takes following actions.

(1) Upon receiving all *apply* messages, it admits those student groups whose parameter *urgent* in its *apply* message is 'true', and update local variable *enrollment*(i). If *enrollment*$(i) >$ *nmb*, then it sends an *abort* message to the sink and stops communication until the third stage. Otherwise, it sends a message *admitted*(*enrollment*(i), college i, group j) to each admitted applicant u_j;
(2) For each student group application with the message *apply*(college i, group j, Δ_{ij}, *urgent* = '*false*'), it does the following:
   (a) Find a student group u_j with the largest Δ_{ij} and $\Delta_{ij} \leqslant$ *nmb*;
   (b) Update local variable *enrollment*$(i) \leftarrow \Delta_{ij}$;
   (c) Send a message *admitted*(*enrollment*(i), college i, group j) to student group u_j;
   (d) Broadcast a message *recruit*(*nmb*, layer l, *enrollment*(i), college i) to all recruiters;
(3) Upon receiving any *abort* message from a student group, it forwards it to the sink and stop communication until the third stage;
(4) Upon receiving *done* messages from all recruiters, it sends a *done*(*enrollment*(i)) message to the sink.

Finally, the sink (the tree root) takes the following actions.

(1) Upon receiving an *abort* message from any $v_i, 1 \leqslant i \leqslant d$, it broadcasts an *abort* message to all student groups through its recruiters along the tree paths;
(2) Upon receiving *done*(*enrollment*(i)) messages from all v_i, it updates the value of *nmb* to be $\max_{1 \leqslant j \leqslant d}\{\text{enrollment}(j)\}, 1 \leqslant i \leqslant d$, and broadcasts a message *termination*(*nmb*) to all recruiters and student groups.

Note that a college admission officer does not need to send a denial message to a student group applicant. This implies that if a student group does not receive an *admitted* message in $2l$ time units and this group can apply to another college in the next round once it receives a new *recruit* message.

5.2.3. The third stage – transfer stage

The admission stage ends with either a *termination* message or an *abort* message from the sink. If it is the former, a student group is assigned to a new parent which may be different from the one prior to the reconnection; otherwise, every student keeps the original parent. In either case, the third stage gives each student group a chance to change its parent node, that is, to transfer to another college if this leads to an improvement on the balance of enrollment, or keep as is. The detailed explanation of this stage consisting of three steps is as follows.

In the first step, each student group $u_j, 1 \leqslant j \leqslant k$, identifies a college i from all communicating colleges that has the smallest value of *enrollment*(i). Suppose u_j currently enrolls in college i_0. Let $\Delta_{ij} =$ *enrollment*$(i) + s_j$. If $i \neq i_0$, then student group u_j sends a transfer request *transfer*(*enrollment*(i_0), college i, group j, Δ_{ij}) to college i.

In the second step, upon having received all *transfer* messages, each admission officer $v_i, 1 \leqslant i \leqslant d$, identifies a transfer message with the largest Δ_{ij} but $\Delta_{ij} \leqslant$ *nmb*, accepts the transfer, updates local variable *enrollment*$(i) \leftarrow$ *enrollment*$(i) + \Delta_{ij}$, and sends a message *admitted*(*enrollment*(i), college i, group j) to u_j.

In the third step, each student group informs its previous college if it has been transferred to another college. Each college admission officer then updates its college enrollment accordingly and notifies the sink. Finally, the sink computes the new *nmb* value and broadcasts it to the entire tree.

For convenience, we will also refer to the `balance-refine algorithm` as 'refinement' for short.

5.3. Correctness and termination

The proposed distributed algorithm for re-connecting two adjacent layers forms the base of algorithm `balance-refine` that consists of $h - 1$ iterations, and the distributed algorithm for reconnecting two adjacent layers is invoked within each iteration.

The correctness of the proposed distributed algorithm is obvious. The rest is to show that it will terminate. Clearly, it will terminate at its first stage - the "start" stage, and second stage - the "transfer" stage. We now show that it will terminate in its last stage - the "admission" stage, too.

Lemma 2. *The admission stage of the distributed algorithm either terminates or aborts within no more than k rounds, where* $k = |V_{l+1}|$ *is the number of student groups with roots at layer* $l + 1, 1 \leqslant l \leqslant h - 1$.

Proof. According to the proposed distributed algorithm, within each round, a college admission officer either receives no applications and so admits no student group, or accepts at least one student group. Therefore, in each round of recruiting, if student group j does not receive any *admitted* message from college i it applied to, then

there must be at least one student group other than group j that has been accepted by college i in this round. Since there are only k student groups, one student group cannot be denied by more than $k-1$ times. Therefore, within k rounds, the *admission* stage will either terminate with all student groups admitted or abort without admitting any student groups. □

Theorem 4. *Given a shortest distance routing tree, the time and message complexities of the* `balance-refine` *algorithm are $O(hn)$ and $O(hn^2)$ respectively, where h is the height of the tree and $n = |N|$ is the number of sensor nodes in the network $M(N \cup \{s\}, L)$.*

Proof. The number of iterations in algorithm `balance-refine` is $h-1$. The time complexity of each iteration is dominated by the *admission* stage. According to Lemma 2, there are at most k rounds and each rounds cost $2l$ time units. The time complexity of the `balance-refine` algorithm thus is $\sum_{i=1}^{h-1} O(i\, n_i) \leqslant h\sum_{i=1}^{h-1} O(n_i) \leqslant O(hn)$, where $n_i = |V_i|$.

The message complexity is dominated by the *admission* stage of the re-connecting procedure. Following Lemma 2, in the *admission* stage of reconnecting layer l and layer $l+1$ ($1 \leqslant l \leqslant h-1$), there are at most k rounds of recruiting, while in each round at most k *apply* messages are sent. Every *apply* message is transmitted to a college admission officer through l-hop relays and then is responded by either an *admitted* message or a new *recruit* message. Thus, the message complexity related to *apply* messages is $O(k^2 l)$. The other message complexity related to *recurit* messages, whenever a college admission officer accepts a student group, the value of *enrollment*(i) increases, it then broadcasts this updated value to all student groups. The message complexity related to *recurit* messages thus is $O(k^2 l)$, too. Therefore, the message complexity within each iteration is $O(k^2 l)$. The number of messages required by the `balance-refine` algorithm is $\sum_{i=1}^{h-1} O(i\, n_i^2) \leqslant h\sum_{i=1}^{h-1} O(n_i^2) \leqslant O(hn^2)$. □

6. Performance evaluation

In this section we evaluate the performance of proposed algorithm in terms of *NMB* and the network lifetime, through experimental simulations. We also investigate the impact of several network parameters, such as network density, network size, and the location of the sink on the performance by the simulations. We refer the proposed algorithm as '`top-down with refinement`' algorithm, since it invoke two algorithms: 'top-down' and 'refinement'.

In a default setting, we consider a sensor network consisting of 100–450 sensors randomly deployed in a 200 m × 200 m square region with the sink randomly deployed in a centered square area of 200/3 m × 200/3 m. We also consider the other cases in which sensors are randomly distributed in a square area with the length of each side ranging from 100 m to 300 m with an increment of 50 m while keeping the node density unchanged. We assume that all sensors are identical, this implies that every sensor has the same initial energy capacity of 0.5 J, an identical transmission radius of $R = 30$ m, and the same sampling rate. Within each session each sensor sends the same amount of sensed data (80 bits) to the sink without any aggregation. As mentioned in Section 2, the dominant energy consumption in wireless networks is the radio communication, we focus on the communication energy consumption by ignoring the other energy consumptions. The amounts of energy consumed by a sensor for transmitting or receiving 1-bit data are computed as follows [9]:

$$e_t = \alpha + \beta R^2, \tag{3}$$

$$e_r = \gamma, \tag{4}$$

where R is the transmission radius, $\alpha = 45 \times 10^{-9}$ J/bit, $\beta = 10 \times 10^{-12}$ J/bit/m^2, and $\gamma = 135 \times 10^{-9}$ J/bit [9]. The value in each figure is the mean of the simulation results of 200 random network topology instances.

To evaluate the performance of the proposed algorithms, an existing 'node-centric' algorithm in [4] is employed as our benchmark, where the 'node-centric' algorithm proceeds iteratively. Initially, the tree contains the only the root, i.e., the sink. Within each iteration, it first selects a branch with the lightest load, and then grafts onto this branch the unassigned/unmarked border node generating the heaviest load. Notice that the 'node-centric' algorithm focuses only on the load balance among the nodes without incorporating the constraint of the distance from each node to the sink. Also, it must be mentioned that the performance comparison between the proposed algorithm and the 'node-centric' algorithm is unfair, because the latter does not guarantee the shortest path routing, and may produce a very 'slim' tree to achieve load balance. The purpose of making use of such a comparison is to observe the difference and the trade-off between load balance and data routing delay in the routing trees delivered by both algorithms.

6.1. A Lower bound

In this subsection, we will establish a lower bound on *NMB* and use this lower bound to evaluate the performance of our algorithm. A trivial lower bound on *NMB* is $|N|/|V_1|$, where $V_1 = Adj(s)$ is the set of nodes forwarding sensed data to the sink directly. Obviously, no solution could be better than this bound. This lower bound however is too loose and may be far below any optimal solution. We observe that if all routing paths must be shortest, some nodes may only be able to reach one particular branch node x in V_1 by this requirement. Based on this observation, for each node $x \in V_1$, we define a set $U(x) = \{v \mid v \in N$, and v can only reach $x \in V_1$ by any shortest path$\}$. Since $NMB(T) \geqslant \max_{x \in V_1}\{|U(x)|\}$ for any routing tree T, $\max_{x \in V_1}\{|U(x)|\}$ can be used as another lower bound on *NMB*. We now generalize this idea further. Let P be any subset of V_1, define $Q(P)$ to be the set of nodes in M that can only reach the nodes in P by any shortest routing, i.e., $Q(P) = \{v \mid v \in N$, and v can only reach one or more nodes in P by a shortest path$\}$. Then, for any routing tree T, $NMB(T) \geqslant \frac{|Q(P)|}{|P|}$. Since we could not exhaust all possible

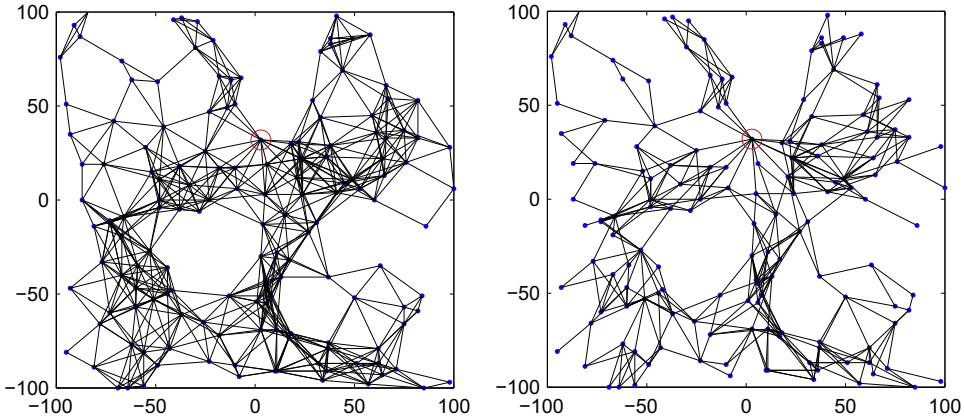
sets of P, we will select a few of them in our simulations as follows.

We first compute the coordinates (θ_v, r_v) of each node $v \in V_1$ in a polar coordinate system with node s being the center, where θ_v is the positive angle from the horizontal line $(s, +\infty)$ and r_v is the distance between s and v. Then, for each $v_k \in V_1, 1 \leqslant k \leqslant d$, we compute a set P_k where $P_k = \{v \mid v \in V_1 \text{ and } (\theta_{v_k} \leqslant \theta_v < \theta_{v_k} + 45^\circ) \text{ and } r_v > \frac{1}{2}R\}$. In other words, P_k is the set of nodes in V_1 that are within the sector of 45° from v_k and $R/2$ away from the sink. Now, another lower bound on *NMB(T)* is derived, which is $\max_{1 \leqslant k \leqslant d}\{\frac{|Q(P_k)|}{|P_k|}\}$. Since the optimal value of *NMB(T)* will above any lower bound, we use the largest one among these three lower bounds as the benchmark to measure how close between the results produced by our algorithms and the achievable optimal results. That is, the following lower bound will be used: $LB(M) = \max\{\frac{|N|}{|V_1|}, \max_{x \in V_1}\{|U(x)|\}, \max_{1 \leqslant k \leqslant d}\{\frac{|Q(P_k)|}{|P_k|}\}\}$.

6.2. Balanced trees delivered by different algorithms

We first observe the differences between balanced routing trees produced by the '`top-down with refinement`' algorithm and by the 'node-centric' algorithm through a concrete example. Fig. 6a shows a randomly generated sensor network and Fig. 6b is the distance graph of the network. The sink is marked by a small circle. Fig. 6c and d are the routing trees produced by the 'node-centric' and our algorithm respectively, from which it can be seen that the tree produced by the 'node-centric' is more balanced among its branches, however the shape of the tree is 'slim', and quite a few nodes in it have long paths to reach the sink. While the tree produced by the proposed algorithm is 'fat', although slightly less balanced, but it allows every node to reach the sink along the shortest path from the node.

As shown by Fig. 7, the average message latency in the routing tree produced by the 'node-centric' algorithm is much longer than that in the routing tree by our proposed algorithm. Moreover, the gap of the average message delay between them increases with the growth of the node density. In fact, the trees produced by the 'top-down' algorithm with or without balance refinement guarantee the routing path from each node to the sink is the shortest one. Also, the one with load-balance refinement has a



(a) The original sensor network M (b) The distance graph of M

(c) The balance tree produced by the node-centric heuristic (d) The balance tree produced by `top-down with refinement`

Fig. 6. The load-balanced trees produced by different algorithms in a sensor network of 150 nodes randomly deployed in a 200 m × 200 m square region.

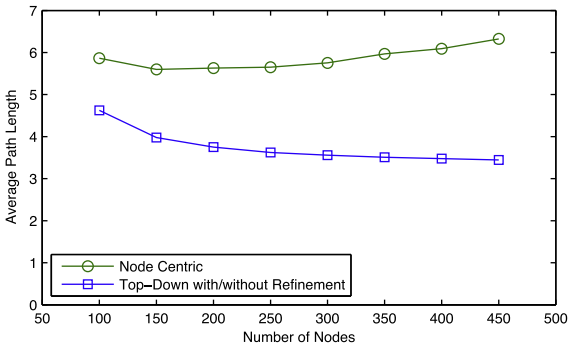


Fig. 7. The average message latency between any node and the sink in a spanning routing tree for a sensor network deployed within a 200 m × 200 m square.

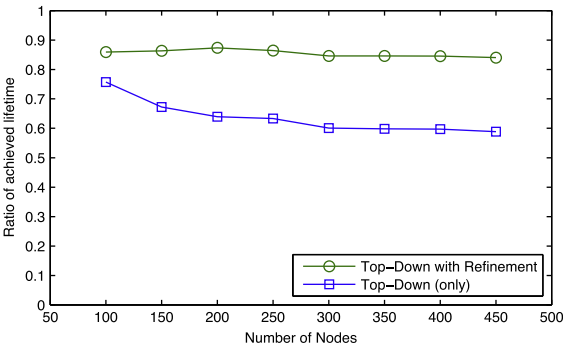


Fig. 9. The ratio of achieved network lifetime over its upper bound in a sensor network deployed in a 200 m × 200 m square region with various node densities.

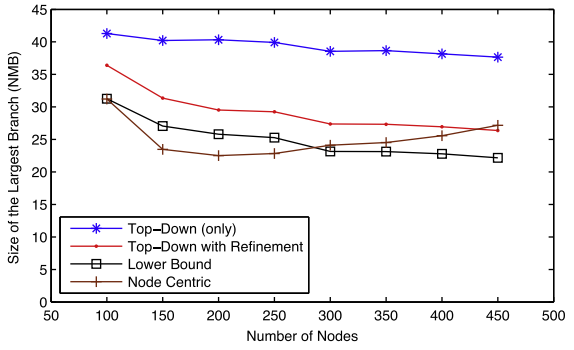


Fig. 8. The size of the largest branch of balanced trees (NMB) delivered by different algorithms for networks deployed in a 200 m × 200 m square with different node densities.

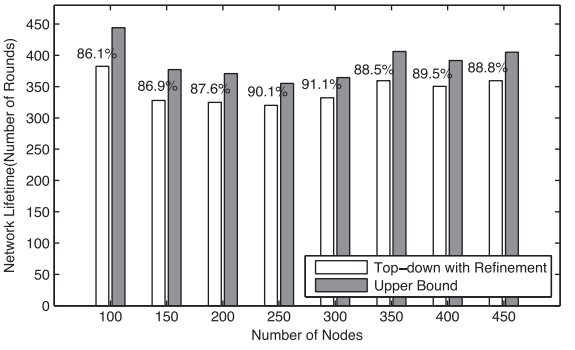


Fig. 10. The network lifetime delivered by the 'top-down' algorithm with balance refinement and its ratio to the upper bound for sensor networks deployed in a 200 m × 200 m square with different node densities, where the sink is located at one of the four corners.

much better load balance, which can be observed from Fig. 8.

Fig. 8 plots the value curves of *NMB* of different algorithms for networks deployed in a 200 m × 200 m square region with different node densities, from which it can be seen that the 'top-down with refinement' algorithm outperforms the 'top-down' algorithm significantly. The curve of the 'top-down with refinement' is parallel to the curve of the lower bound, which means its performance is stable. Since the curve of the optimal solution lies inside the gap, our solution is less than six nodes from the optimal solution. In contrast, the *NMB* obtained by the 'node-centric' algorithm tends to be worse with the growth of network size. Notice that the value of *NMB(T)* of a tree *T* delivered by the 'node-centric' algorithm may even below the lower bound, because the tree is not necessarily to be a shortest one. However, as seen from Fig. 8, *NMB(T)* of the 'node-centric' algorithm becomes worsen when the node density reaches 300 or above.

6.3. Evaluation of network lifetime of different routing trees by different algorithms

We then study the performance of the 'top-down' algorithm and the 'top-down' algorithm with balance refinement against an upper bound on the maximum network lifetime defined in Section 2.

Since the network lifetime is inversely proportional to the value of *NMB*, then a lower bound of *NMB* can be converted to an upper bound on the maximum network lifetime, and this upper bound is obviously larger than the maximum network lifetime. Table 1 illustrates the net-

Table 1
The network lifetime delivered by different algorithms in a deployed network in a 200 m × 200 m square region with various node densities.

Algorithm	Number of nodes							
	100	150	200	250	300	350	400	450
Top down	800	822	819	828	857	855	866	878
Top down with refinement	908	1054	1119	1130	1207	1209	1227	1253
Upper bound	1057	1222	1282	1307	1427	1430	1451	1491

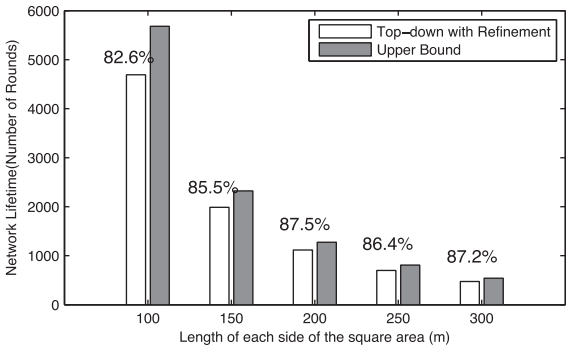


Fig. 11. The network lifetime delivered by the 'top-down with refinement' algorithm and its ratio to the upper bound of the optimal network lifetime for sensor networks deployed in a square with various edge lengths while the node density $1/200(\text{nodes}/\text{m}^2)$ is fixed.

work lifetimes achieved by different algorithms in the same sensor network, in comparison with an upper bound on the maximum network lifetime.

Fig. 9 demonstrates that the 'top-down' algorithm with load balance refinement can achieve a network lifetime no less than 85% of the maximum possible one (the upper bound on the maximum network lifetime). As the best achievable network lifetime for the network is in the middle of the gap, the proposed algorithm may achieve 92% of the maximum network lifetime.

Extensive simulations have also been conducted to evaluate the network lifetime when the sink is located at one of the four corners, rather than at the center of the monitoring area. Because of symmetry, we assume that the sink is located at the most upright corner in our experiments. Fig. 10 shows that the network lifetime delivered by the 'top-down' algorithm with balance refinement achieves 86% of the upper bound of the maximum network lifetime.

We finally evaluate the performance of our proposed algorithm by varying the network size while keeping the node density fixed at $1/200(\text{node}/\text{m}^2)$. The performance of the 'top-down' algorithm with load-balance refinement is shown in Fig. 11, from which it can be seen that with the increase of the square size, the achieved network lifetime decreases. The figure also shows that the 'top-down' algorithm with load-balance refinement always delivers a solution that is very close to the upper bound of the maximum network lifetime, i.e., the network lifetime delivered by it is no less than 82% of the optimal one.

7. Conclusion and future work

In this paper we studied the network lifetime maximization problem for time-sensitive data gathering applications, through constructing a load-balanced shortest distance routing tree. We first formulated the problem as an optimization problem and showed its NP-hardness. We then devised a novel 'top-down' algorithm that constructs a balanced shortest distance routing tree layer by layer, and each layer extension is optimal from the current tree. A distributed implementation of the 'top-down' algorithm has also been given. To further improve the performance of the 'top-down' algorithm, we proposed a 'balance-refine' distributed algorithm. Finally, we conducted extensive simulations to evaluate the performance of the proposed algorithms. The simulation results have shown that the network lifetime delivered by the 'top-down' algorithm with balance refinement is no less than 85% of the optimal network lifetime.

There are several interesting directions for future works. For example, designing an approximation algorithms to construct the balanced shortest path (routing) tree, which may have potential applications in many other research area; considering building the balanced shortest routing tree for sensor networks in which sensors are of non-uniform energy levels; relaxation of shortest path constraints, that is sacrificing message delays from a small portion of nodes to improve the balance of the routing tree and time complexity of the algorithm.

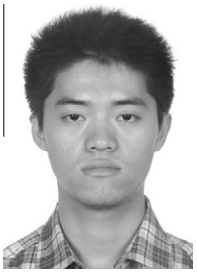
Acknowledgments

The authors would like to thank the anonymous referees and the editor for their constructive comments and valuable suggestions which have helped improve the quality and presentation of the paper. The work of Feng Shan was done during his visit to University of Missouri-Kansas City, USA. His research is supported by National Key Basic Research Program of China under Grants No. 2010CB328104, China Scholarship Council, NSFC under Grants No. 61070161, No. 61003257, No. 61202449, No. 61272531, No. 61272054, China National Key Technology R&D Program under Grants No. 2010BAI88B03 and No. 2011BAK21B02, SRFDP under Grants No. 20110092130002, China National Science and Technology Major Project under Grants No. 2010ZX01044-001-001. Jiangsu Provincial Key Laboratory of Network and Information Security under Grants No. BM2003201.

References

[1] B. Awerbuch, R.G. Gallager, A new distributed algorithm to find breadth first search trees, IEEE Transactions on Information Theory IT-33 (1987) 315–322.
[2] C. Buragohain, D. Agrawal, S. Suri, Power aware routing for sensor databases, in: Proceedings of INFOCOM'05, IEEE, 2005.
[3] R. Cristescu, B. Beferull-Lonzano, M. Vetterli, On network correlated data gathering, in: Proceedings of INFOCOM'04, IEEE, 2004.
[4] H. Dai, R. Han, A node-centric load balancing algorithm for wireless sensor networks, in: Proceedings of GLOBECOM'03, IEEE, 2003.
[5] M. Fürer, B. Raghavachari, Approximate the minimum-degree Steiner tree to within one of optimal, Journal of Algorithms 17 (1994) 409–423.
[6] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman Company, NY, 1979.
[7] A. Goel, D. Estrin, Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk, in: Proceedings of SODA'03, ACM-SIAM, 2003.
[8] A.V. Goldberg, R.E. Tarjan, A new approach to the maximum-flow problem, Journal of ACM 35 (1988) 921–940.
[9] W. Heinzelman, Application-Specific Protocol Architectures for Wireless Networks, Ph.D. Thesis, Massachusetts Institute of Technology, 2000.

[10] P.H. Hsiao, A. Hwang, H.T. Kung, D. Vlah, Load-balancing routing for wireless access networks, in: Proceedings of INFOCOM'01, IEEE, 2001.
[11] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, Directed diffusion for wireless sensor networking, IEEE/ACM Transactions on Networking 11 (2003) 2–16.
[12] C. Intanagonwiwat, D. Estrin, R. Govindan, J. Heidemann, Impact of network density on data aggregation in wireless sensor networking, in: Proceedings of ICDCS'02, IEEE, 2002.
[13] K. Kalpakis, K. Dasgupta, P. Namjoshi, Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks, Computer Networks 42 (2003) 697–716.
[14] S. Khuller, B. Raghavachar, N. Young, Balancing minimum spanning and shortest path trees, in: Proceedings of SODA'93, ACM-SIAM, 1993.
[15] W. Liang, Y. Liu, Online data gathering for maximizing network lifetime in sensor networks, IEEE Transactions on Mobile Computing 6 (2007) 2–11.
[16] J. Liang, J. Wang, J. Cao, J. Chen, M. Lu, An efficient algorithm for constructing maximum lifetime tree for data gathering without aggregation in wireless sensor networks, in: Proceedings of INFOCOM'10, IEEE, 2010.
[17] S. Lindsey, C.S. Raghavendra, PEGASIS: power-efficient gathering in sensor information systems, in: Proceedings of Aerospace Conference, IEEE, 2002.
[18] S. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, TAG: a Tiny AGgregation service for ad-hoc sensor networks, in: Proceedings of OSDI'02, ACM, 2002.
[19] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, Wireless sensor networks for habitat monitoring, in: Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications, ACM, 2002.
[20] G.J. Pottie, Wireless sensor networks, in: Proceedings of Information Theory Workshop, IEEE, 1998.
[21] P. von Richenbach, R. Wattenhofer, Gathering correlated data in sensor networks, in: Proceedings of DIALM-POMC, ACM, 2004.
[22] M.A. Sharaf, J. Beaver, A. Labrinidis, P.K. Chrysanthis, Balancing energy efficiency and quality of aggregate data in sensor networks. Journal of VLDB (2004).
[23] A. Singh, M. Woo, C.S. Raghavendra, Power-aware routing in mobile ad hoc networks, in: Proceedings of MobiCom, ACM/IEEE, 1998.
[24] Y. Wu, Z. Mao, S. Fahmy, N.B. Shroff, Constructing maximum-lifetime data gathering forests in sensor networks, IEEE/ACM Transactions on Networking 18 (2010) 1571–1584.
[25] Y. Wu, S. Fahmy, N.B. Shroff, On the construction of a maximum-lifetime data gathering tree in sensor networks: NP-completeness and approximation algorithm, in: Proceedings of INFOCOM'08, IEEE, 2008.
[26] T. Yan, Y. Bi, L. Sun, H. Zhu, Probability based dynamic load-balancing tree algorithm for wireless sensor networks, in: Proceedings of ICCNMC 05, LNCS, 2005.



Feng Shan received his B.S. degree in Computer Science from Hohai University, Nanjing, China, in 2008. He is currently pursuing the Ph.D. degree in computer science and engineering at Southeast University, Nanjing, China. He joined the Department of Computer Networking, University of Missouri-Kansas City, Kansas City, MO, United States, from 2010 to 2012 as a visiting scholar. His research interests are in the areas of Wireless Sensor Network, algorithm, and wireless multi-hop networks.

Weifa Liang (M'99–SM'01) received the PhD degree from the Australian National University in 1998, the ME degree from the University of Science and Technology of China in 1989, and the BSc degree from Wuhan University, China in 1984, all in computer science. He is currently an Associate Professor in the Research School of Computer Science at The Australian National University. His research interests include design and analysis of energy-efficient routing protocols for wireless ad hoc and sensor networks, information processing in wireless sensor networks, cloud computing, design and analysis of parallel and distributed algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE.

Jun Luo received his M.S. degree in Software Engineering from National University of Defense Technology, China, in 1989, and the B.S. degree in Computer Science from Wuhan university, China, in 1984. He is currently a full professor at School of Computer in National University of Defense Technology, China. His research interests are in ad hoc and sensor networks, design of energy-efficient protocols for wireless networks and operating systems.

Xiaojun Shen received his Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, in 1989. He received his M.S. degree in computer science from the Nanjing University of Science and Technology, China, in 1982, and his B.S. degree in numerical analysis from the Tsinghua University, Beijing, China, in 1968. He is currently a professor in the School of Computing and Engineering at the University of Missouri-Kansas City. He is a senior IEEE member. His current research interests include computer algorithms and computer networking with focus on routing and scheduling.