A Lifetime-Preserving and Delay-Constrained Data Gathering Tree for Unreliable Sensor Networks

Yanjun Li[1,2], Yueyun Shen[1] and Kaikai Chi[1]
[1] School of Computer Science and Technology, Zhejiang University of Technology
Hangzhou 310023 - China
[2] Zhejiang Provincial Key Laboratory of Information Network Technology
Hangzhou 310027 - China
*Corresponding author: Yanjun Li
[e-mail: {yjli, kkchi}@zjut.edu.cn, yueyunshen@gmail.com]

Received March 15, 2012; revised July 5, 2012; revised August 29, 2012; accepted December 2, 2012; published December 27, 2012

Abstract

A tree routing structure is often adopted for many-to-one data gathering and aggregation in sensor networks. For real-time scenarios, considering lossy wireless links, it is an important issue how to construct a maximum-lifetime data gathering tree with delay constraint. In this work, we study the problem of lifetime-preserving and delay-constrained tree construction in unreliable sensor networks. We prove that the problem is NP-complete. A greedy approximation algorithm is proposed. We use expected transmissions count (ETX) as the link quality indicator, as well as a measure of delay. Our algorithm starts from an arbitrary least ETX tree, and iteratively adjusts the hierarchy of the tree to reduce the load on bottleneck nodes by pruning and grafting its sub-tree. The complexity of the proposed algorithm is $O(N^4)$. Finally, extensive simulations are carried out to verify our approach. Simulation results show that our algorithm provides longer lifetime in various situations compared to existing data gathering schemes.

Keywords: network lifetime, delay constraint, data gathering tree, aggregation, sensor networks

This work was supported by China NSF (No.61003264, 61001126), the open project of State Key Lab. of Industrial Control Technology under Grant No. ICT1110 and the open project of Zhejiang Provincial Key Lab. of Information Network Technology.

http://dx.doi.org/10.3837/tiis.2012.12.011

1. Introduction

Data gathering is a basic operation for many sensor network applications. In most cases, the sink only requires information aggregated by different nodes [1][2], e.g., average humidity, maximum pressure, top-*k* temperature, an indication whether a target is present, etc. In such cases, intermediate nodes perform respective functional computations on received data from multiple sources, which is known as a viable way to reduce redundant transmissions. Since sensor nodes are typically battery-powered, it is crucial to conserve energy so that the network lifetime can be prolonged. On the other hand, in some real-time applications, such as fire monitoring, gas or radiation leakage tracking, information timeliness is also important. Out-of-date information is irrelevant and may even lead to negative effects to the system. It is thus required that the data sensed from the sources should be transmitted to the sink within predestined delay constraints. Therefore, it is worth investigating how to collect aggregated data in an energy efficient and real-time manner.

The tree-based topology is often adopted for data gathering because of its simplicity. It saves the cost of maintaining a routing table at each node and it does not require the location of each node. Here the network lifetime is defined as the time until the first node is drained of its energy. It has been proved in [3] that finding a maximum lifetime tree from all feasible spanning trees is NP-complete, and an approximation algorithm for constructing a sub-optimal tree is thereby proposed. Luo et al. [4] argue that this solution may result in intolerable delay if the tree is deep and it is thus not suitable for time-critical applications. Therefore, they study the problem of selecting a maximum lifetime tree from the set of shortest path (hop) trees (SPT). However, their solution holds only when the link quality in the network is perfect. In practice, on the contrary, low-power wireless links suffer from unexpected packet loss. With an unreliable link, a transmission has to be repeated if previous transmissions are unsuccessful. It is most likely that the shortest path consists of long links, which tend to have unsatisfying link qualities. As a result, the shortest path does not necessarily indicate a guaranteed delay and may even costs more energy in retransmissions.

Motivated by the above mentioned limitations in previous work, we study the problem of delay-constrained tree construction for preserving network lifetime in unreliable sensor networks. We prove that the problem is NP-complete. Consider that the links are lossy and unreliable, we use expected transmission counts (ETX) as the link quality indicator, as well as a measure of delay. End-to-end transmission count is a commonly used metric to represent end-to-end delay, as a larger transmission count leads to a longer delay [5]. Our algorithm starts from an arbitrary least ETX tree, and iteratively adjusts the hierarchy of the tree to reduce the load on bottleneck nodes. Meanwhile, the adjustment is only allowed with the maintenance of end-to-end delay constraint. Extensive simulations are carried out to verify our approach. Simulation results show that our algorithm provides longer lifetime compared to exiting popular data gathering schemes.

The remainder of the paper is organized as follows. Section 2 discusses related work on data gathering protocols. Section 3 describes the system model and formulates the problem. Section 4 elaborates on our tree construction algorithm and analyzes its time complexity. Simulations are conducted in Section 5 to evaluate the performance. Finally, Section 6 concludes the paper.

2. Related Work

Research on data gathering protocols has been extensively investigated in the literature. Here we briefly review a handful of the most related work to our study.

A class of data gathering protocols in sensor networks is based on geographic forwarding schemes. Some of them aim at supporting end-to-end delay guarantee, such as SPEED [6], MMSPEED [7], and THVR [8] etc. They try to achieve desired end-to-end delays by enforcing a uniform forwarding velocity throughout the network. However, these protocols need to work with localization techniques and they do not explicitly consider the issue of network lifetime preserving. A few aim at prolonging network lifetime solely. For example, Lim et. al [9] propose a geographic forwarding scheme to improve lifetime by considering the residual energy of neighbors in deciding the next-hop and removing undesirable geographical neighbors.

Cluster-based structure is also popular for data gathering. Typical cluster-based protocols include LEACH [10], HEED [11] etc. Cluster-based protocols are easy to be implemented, but they have some disadvantages such as uncontrollable cluster size and heavy load on cluster heads. A few data gathering protocols are chain-based, e.g., PEGASIS [12], in which a chain among the sensor nodes is formed so that each node communicates with a close neighbor using low power radio. In this way nodes conserve their energy, but a long chain would cause intolerable delay.

Tree topology is widely used for data gathering. Some tree-based protocols aim at minimizing energy consumption, such as PEDAP and PEDAP-PA [13], which use a heuristic to assign weights to links and find a minimum spanning tree (MST) rooted at the sink node in terms of total transmission energy consumption. PEDAP and PEDAP-PA do reduce the energy consumption but makes no contribution to lifetime preserving, as some critical nodes are over loaded and run out of energy prematurely. Some protocols thus propose to construct a tree with maximum lifetime. Wu et al. [3] prove that constructing an arbitrary aggregation tree with maximum lifetime is NP-hard, and they propose an approximation algorithm. Luo et al. [4] argue that this solution may result in intolerable delay if the tree is deep and it is thus not suitable for time-critical applications. They study the problem of selecting a maximum lifetime tree from shortest path trees, on the intuition that shorter path usually has shorter delay.

However, most of the above work assumes perfect link quality. In effect, recent experimental studies have shown that real deployment of sensor networks has "transitional region" with highly unreliable links [14]. The shortest path from a source to the sink tends to be comprised of a few long sing-hop links falling into the "transitional region" with moderate or bad link qualities. Therefore, link quality has to be considered in the protocol design if an end-to-end delay guarantee is required. Otherwise the designed protocol will possibly fail to reach an expected performance on real platforms. There are a few data gathering protocols taking the lossy link properties into consideration. CTP [15] is a well-known protocol which provides best-effort anycast routes to the sink. Liang et al. [16] use path ETX to denote the path length and construct a shortest path tree for each channel. These works are based on real systems, but they do not explicitly consider the network lifetime, neither the end-to-end delay. It is worth noting that none of the above work considers both network lifetime and delay constraint in an unreliable network environment. This gap has been filled by our work in this paper. We study the problem of finding a lifetime-preserving tree with delay constraint in unreliable sensor networks. Our algorithm starts from an arbitrary least ETX tree as in [16], and then iteratively adjusts the hierarchy of the tree to reduce the traffic load on bottleneck nodes by pruning and grafting its subtree to another node. Meanwhile, end-to-end delay constraint has to be satisfied after the adjustment. Finally a tree with maximized lifetime while satisfying the delay constraint is obtained.

3. System Model and Problem Formulation

3.1 System Model

We consider the network as an undirected graph $G(V, E)$, in which $V=\{0, 1, 2, ..., N\}$ denotes the set of $N+1$ sensor nodes with node 0 referring to the sink, and E is the set of edges in the network. We define that there is an edge between two nodes i and j if and only if the link Packet Reception Rate (PRR) $q(i, j)$ is no less than a given threshold $q_{t,}$ namely, $(i, j) \in E$ if and only if $q(i, j) \geq q_{t}$. Each node i has different initial energy $e(i)$ while the sink has infinite energy $e(0)=+\infty$, as generally assumed in other literature [17]. The sensor network is progressed in rounds, and in each round each node generates a k-bit data packet, which should be aggregated and sent to the sink within the delay constraint D_c. The end-to-end delay of tree T, denoted by $D(T)$, is defined as the largest delay for a node in tree T to deliver a packet to the sink. The amount of energy required to transmit and receive a k-bit data packet is e_t and e_r respectively.

We follow common assumptions in [3] and [4]. First, the network is at least 1-connected. It is essential for every node to be part of the data gathering tree. Nodes without connectivity are unable to provide data to the base station and are thus of no use. Second, we assume the network has applied TDMA-based MAC to avoid collisions and overhearing, hence only transmission and reception energy is recorded. Other energy consumptions, such as computation and switching cost are neglected. Third, each node aggregates or fuses the received multiple k -bit data into a single k-bit ongoing packet. Besides, we do not explicity consider the ACK loss and ignore the energy spent on sending and receiving ACKs. These assumptions are reasonable for the following reasons. First, the link PRR $q(i, j)$ can be calculated as the average of two-way PRRs, i.e. $q(i, j) = (q_{i \to j} + q_{j \to i})/2$. In this sense, the ACK loss can be assumed to have been taken into account. Furthermore, it has been observed through testbed experiments that the reliability of ACK is still high even with interferences and unreliable links [18]. This is due to the small size of ACK and prompt transmission right after the reception. Second, the energy spent on sending and receiving ACKs is relatively small compared with that for a common packet, most likely in different order of magnitude.

The notations used in this paper are listed in **Table 1**. For $(i, j) \in E$ (i is the transmitter and j the receiver), the link ETX is the inverse of the PRR value $q(i,j)$, namely $1/q(i,j)$, denoted by $E_t(i, j)$. If there are multi-hops between node i and node j, the end-to-end path ETX is computed by the sum of each single-hop ETX, and it is tree-dependent, denoted by $E_t(T, i, j)$. Noting that there is a bound for the retransmission count over a lossy link, e.g., the IEEE 802.15.4 standard [19] introduces the parameter *macMaxFrameRetries* to hold the maximum number of transmissions. Similarly, we introduce a parameter N_t to denote the maximum transmission count in each hop. The average number of transmissions over link (i, j), denoted by $X_t(i, j)$, should be

$$X_t(i, j) = \lceil \min(E_t(i, j), N_t) \rceil \tag{1}$$

For tractability of the problem, we choose q_t such that $q_t \geq 1/N_t$. Therefore, $X_t(i, j)$ can be simplified to

$$X_t(i, j) = \lceil E_t(i, j) \rceil = \lceil 1/q(i, j) \rceil \tag{2}$$

where the notation $\lceil \;\; \rceil$ represents the ceil function.

The energy consumption of an intermediate node in a round is the sum of energy depletion for successfully transmitting a packet to its parent and that for receiving all packets from its children. Let $C(T,i)$ denotes the child node set of node i in tree T, in each round the amount of energy node i consumes is equal to $e_t X_t(i, p(T,i)) + e_r \sum_{j \in C(T,i)} X_t(j,i)$, where $p(T, i)$ is the parent of node i in tree T. The expected lifetime of the node i in tree T, denoted by $E_l(T,i)$, is thus the total number of rounds the node can sustain. We have

$$E_l(T,i) = \left\lfloor \frac{e(i)}{e_t X_t(i, p(T,i)) + e_r \sum_{j \in C(T,i)} X_t(j,i)} \right\rfloor \tag{3}$$

where the notion $\lfloor \;\; \rfloor$ represents the floor function.

Hitherto, there are various definitions for "network lifetime" [20], among which the definition as the time until the first node depletes its energy has been widely used in the literature [3][4]. Here we also adopt this definition for the following reasons: first, we are motivated by critical applications with strict coverage requirements. For these applications, even single node failure will cause loss of coverage. Second, considering an aggregation tree, the failure of a parent node will prevent all its subtree nodes from transmitting aggregated information, which will greatly impair the network function. By this definition, the network lifetime under a tree T is

$$E_l(T) = \min_{i \in V} E_l(T,i) \tag{4}$$

In a given tree T, except for the sink, we refer to a non-leaf node as a relay node and let each relay node use TDMA scheme to collect data packets from its child nodes. The collection duration of each relay node is measured by constant frames, denoted by σ, and each frame has N_s time slots, which can be determined by the largest node degree in the network. In each frame, each slot is assigned to a respective child node, i.e., the child node is only allowed to transmit its data packet in its respective slot. If one transmission fails, the child node has to wait until its slot in the next frame. Thus, the expected delay for link $(i, j) \in E$ (i is the transmitter and j the receiver) is

$$D(i, j) = \sigma N_s X_t(i, j) \tag{5}$$

where $X_t(i,j)$ is calculated by Eq.(2), both σ and N_s are constants once the network is launched.

Obviously, the end-to-end delay of tree T, denoted by $D(T)$, is the maximum delay among the end-to-end delays from the leaf nodes (rather than relay nodes) to the sink. Hence, $D(T)$ can be obtained through the following Maximum Delay Finding Algorithm (MDFA).

Algorithm 1: Maximum Delay Finding Algorithm

```
Input: Tree $T$, ETX value for each link $(i, j) \in T$
Output: $D(T)$
01: $D_{\max} \leftarrow 0, TempD \leftarrow 0$;
02: while $V_L$ is not empty do
03:     take a leaf node $i$ in $V_L$, $V_L \leftarrow V_L - \{i\}$;
04:     $j \leftarrow p(T, i)$, compute $X_t(i, j)$ by Eq. (2);
05:     $TempD \leftarrow TempD + \sigma N_s X_t(i, j)$
06:     if $j \neq 0$ (i.e., node $j$ is not the sink) then
07:         $i \leftarrow j$, goto line 4;
08:     end
09:     if $TempD > D_{max}$ then
10:         $D_{max} \leftarrow TempD$, $TempD \leftarrow 0$;
11:     end
12: end
13: output $D(T) \leftarrow D_{\max}$.
```

Table 1. Teminology

Symbols	Definitions
$G(V, E)$	The graph of the network, V is the set of nodes and E the set of edges.
T	The aggregation tree constructed from $G(V, E)$ with the sink being the root.
e_t	Energy consumption for transmitting a k-bit packet
e_r	Energy consumption for receiving a k-bit packet
$e(i)$	Initial energy of node i
$q(i, j)$	The link PRR between node i and node j
q_t	The PRR threshold, iif $q(i, j) \geq q_t$, there is an edge between node i and node j
$E_t(i, j)$	The ETX over link (i, j)
$E_t(T, i, j)$	The multi-hop ETX between node i and node j in tree T
N_t	Maximum transmission count
$X_t(i, j)$	The average number of transmissions over link (i, j)
$E_l(T, i)$	Expected lifetime of node i in tree T, i.e., in tree T, node i is expected to sustain $E_l(T, i)$ rounds
$E_l(T)$	Expected network lifetime of a tree T, i.e., the tree T is expected to sustain $E_l(T)$ rounds
$C(T, i)$	The set of node i's children in tree T
$P(i)$	The set of node i's candidate parents
$p(T, i)$	The parent of node i in tree T
D_c	Delay constraint
$D(T)$	End-to-end delay of tree T

3.2 Problem Formulation

Based on the system model and assumptions, we formulate the problem as follow.

Problem 3.2 (**MLDCT**). *Given a network $G(V, E)$ composed of N sensing nodes and one sink node. Each node has an initial energy $e(i)$, $i=0,1,2,...,N$, and periodically generates a packet. The sink node has infinite power. An aggregation tree is constructed to collect data from all the sensors. Data are required to be delivered to the sink within a given delay constraint D_c. The problem is to find a Maximum Lifetime Delay-Constrained Tree T_{opt} in the graph. We call it MLDCT problem for short. Formally, We have*

$$T_{opt} = \arg\max_{T} E_l(T) = \arg\max_{T} \min_{i\in V} E_l(T,i), \\ \text{s.t. } D(T_{opt}) < D_c \tag{6}$$

Proposition 3.2. *MLDCT problem is NP-complete*.

Proof. Clearly, the problem belongs to NP, since given a graph and a tree on it, it is easy to verify whether the tree achieves the maximized lifetime. It has been proved in [2] that constructing a maximum-lifetime tree from any network $G(V, E)$ is a NP-complete problem. We refer to it as MLT problem for short. In effect, MLT problem is a special case of MLDCT problem. Let the link PRR $q(i, j)$ of each edge (i, j) equal to 1, and the delay constraint D_c be $+\infty$, the MLDCT problem can be converted to the MLT problem.

Since MLDCT problem is NP-complete, we propose a heuristic solution in Section 4 to solve it.

4. Algorithm Design

In order to get the smallest end-to-end delay of a tree, we start from an arbitrary least ETX tree (LET), in which the end-to-end delay measured by path ETX from each node to the sink is minimum. Then, we recursively adjust the hierarchy of the tree to reduce the load on bottleneck nodes until no improvement can be further reached. The adjustment is allowed on the premise that the end-to-end delay can satisfy the delay constraint.

4.1 LET Construction

We apply Dijkstra's shortest path algorithm [21] to construct the LET rooted at the sink. Each node stores its least ETX to the sink. ETX can be considered as a reflection of end-to-end delay. If the link quality is perfect, the LET is equivalent to a SPT (the path length is denoted by hop counts to the sink). Note that there may be multiple LET for a given G, and we just choose an arbitrary one. The reason for LET construction is to minimize the delay of the tree.

4.2 Recursive Adjustment

After the LET construction, we have to recursively adjust the hierarchy of the tree until no improvement in the network lifetime can be further reached. Before adjusting the hierarchy of the tree, the lifetime of each node need be computed to find the bottleneck nodes so that the adjustment can be narrowly targeted. Given a tree T, the expected lifetime for each node in the tree is computed by Eq. (3). We define the two nodes with the minimum and the second minimum lifetime as the bottleneck node b and sub-bottleneck node s, respectively. Other nodes are defined as common nodes.

Definition 1 (**bottleneck node**). *Node b is the bottleneck node of tree T if it satisfies*:

$$b = \arg\min_{i\in V}(E_l(T,i)) \tag{7}$$

Definition 2 (**sub-bottleneck node**). *Node b is the bottleneck node of tree T by Definition 1, Node s is the sub-bottleneck node of tree T if it satisfies*:

$$s = \arg \min_{i \in V - \{b\}} (E_l(T, i)) \tag{8}$$

We sort the ETX values of the links from all *b*'s children to *b* in descending order and check them one by one to see whether any correspondent child node *c* can find a type I eligible edge (*c*,*u*) added to *T* to substitute its original edge (*c*,*b*), where type I eligible edge is defined as follows:

Definition 3 (**type I eligible edge**). *The edge (c,u)$\in$E is type I eligible edge if it satisfies the following conditions*:

- *The new tree T' generated by pruning the edge* (*c*,*b*) *and grafting* (*c*,*u*) *still satisfies* $D(T')<D_c$;
- *Neither node c nor u is the bottleneck or sub-bottleneck node in the new tree T'.*

If a type I eligible edge can be found and respective edge pruning and grafting have been conducted, there will be two possible cases:

1) The expected lifetime of the original bottleneck node *b* is increased but not enough to turn it into the sub-bottleneck or common node. In this case, we continue to check *b*'s remaining children, until all the children have been checked. See Algorithm 2, lines 7-8.
2) The expected lifetime of the original bottleneck node *b* is increased and it is no longer a bottleneck node. Then a new bottleneck node *b*' can be found, and we follow the same steps as we did to node *b*. See Algorithm 2, lines 9-10.

If after checking all its children, the bottleneck node *b* cannot turn into a sub-bottleneck node or common node still, we will resort to checking *b*'s candidate parents except its current parent p_c. We sort the ETX values of the links from *b* to all its candidate parents in ascending order and check one by one whether any parent node *p* has better link quality to *b*, and they can form type II eligible edge (*b*,*p*) defined as follows.

Definition 4 (**type II eligible edge**). *The edge (b,p)$\in$E is type II eligible edge if it satisfies the following conditions*:

- *The new tree T' generated by pruning the edge* (*b*, p_c) *and grafting* (*b*, *p*) *still satisfies* $D(T')<D_c$;
- Node *p* is neither the bottleneck nor the sub-bottleneck node in the new tree *T'*.

If a type II eligible edge can be found and respective edge pruning and grafting have been conducted, there will be two possible cases:

1) Node *b* is still the bottleneck node, which implies that the current tree cannot be further adjusted, because the remaining links to be checked are with even larger ETX. Then we output the current tree. See Algorithm 2, line 17-18.
2) Node *b* can get rid of the role of bottleneck node, and the checking process terminates. Then a new bottleneck node *b*' can be found, and we follow the same steps as we did to node *b*. See Algorithm 2, line 19-20.

In summary, based on the above process, we recursively adjust the traffic around the bottleneck nodes and gradually improve the lifetime of the sensor network. Finally a greedy approximate solution to MLDCT problem, namely, T_{LEG} (Least ETX tree with Greedy adjustment) can be obtained.

The procedure of the algorithm in the form of pseudo-code is summarized in the following Algorithm 2.

Algorithm 2: Greedy Algorithm for MLDCT Problem

```
Input: graph $G(V, E)$, $e(i)$ for each $i \in V$, ETX value for each edge $(i, j) \in E$, delay
constraint $Dc$.
Output: $T_{LEG}$, a greedy solution to MLDCT problem
01: construct an arbitrary LET $T$ in $G$;
02: for $i \leftarrow 1$ to $k$
03: while a new bottleneck node $b$ is generated in tree $T$ do
          sort $C(T, b)$ by their ETX values to $b$ in descending order, denoted by $C_d(T, b)$;
04:     foreach node $c \in C_d(T, b)$ do
05:         find a type I eligible edge $(c,u) \in E$, $(c,u) \notin T$;
06:         prune $(c,b)$ and graft $(c,u)$, generating a new tree $T'$;
07:         if the bottleneck node of $T'$ is still $b$ then
08:             continue;
09:         else
10:             goto line 2;
11:         end
12:     end
13:     sort $P(b)$ by the ETX values from $b$ to them in ascending order, denoted by $P_a(b)$;
14:     foreach node $p \in P_a(b)$ do
15:         find a type II eligible edge $(b, p) \in E$, $(b, p) \notin T$;
16:         prune $(b, p(T, b))$ and graft $(b, p)$, generating a new tree $T'$;
17:         if the bottleneck node of $T'$ is still $b$ then
18:             goto line 24;
19:         else
20:             break;
21:         end
22:     end
23: end
24: output the current tree, namely $T_{LEG}$.
```

By example, we illustrate the adjustment operation on the sensor network with the topology shown in **Fig. 1** (a), where solid lines correspond to edges in the tree, and dotted lines denote edges not in the tree. The ETX value of each edge is also indicated. We assume e_t=2, e_r=1, D_c=5σN_s. We can make N_s equal to 5 in this example as the largest node degree in the network is 5. Since σ and N_s are constants and the end-to-end delay is also an integral multiple of σN_s, we can simply let D_c=5 (units) instead of D_c=5σN_s. The initial energy of the nodes are as follows: $e(1)=e(4)=e(7)=2000$, $e(2)=e(8)=3000$, $e(3)=e(5)=e(6)=1500$. According to Eq. (3), node 4 is the bottleneck node with $E_l(T,4)$=266, and node 3 is the sub-bottleneck node with E_l $(T,3)$=300. For node 8, no type I eligible edge can be found. For node 7, the edge (7,6) is type I eligible edge according to Definition 3. By pruning the edge (7,4) and grafting the edge (7,6), a new tree T' is generated as shown in **Fig. 1**(b), which has a longer lifetime than T. It is because after the adjustment, the bottleneck node 4 is substituted by node 3 with E_l $(T',3)$=300 and node 4 becomes sub-bottleneck node with $E_l(T',4)$=333. In other words, the network lifetime is improved by an amount of E_l $(T',3)$- E_l $(T,4)$. As node 3 is the new bottleneck node in T' and it has no child nodes, we turn to check its candidate parent nodes and find the type II eligible edge. According to Definition 4, edge (3,2) is the first type II eligible edge. By pruning the edge (3,0) and grafting the edge (3,2), T_{LEG} is found, whose lifetime $E_l(T_{LEG})$=333 cannot be further improved. Hereto the adjustment operation terminates.

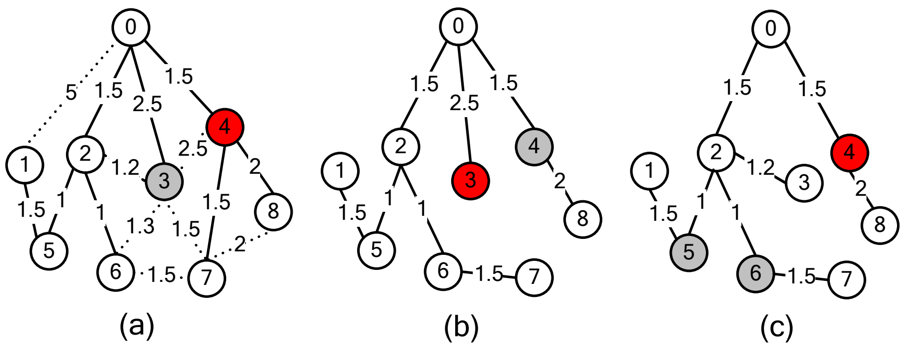


Fig. 1. An illustration of tree adjustment

4.3 Time Complexity Analysis

In this subsection we analyze the complexity of our algorithm, which is measured by the number of operations such as comparisons, calculations, etc. First, Dijkstra's algorithm is used to construct a least ETX tree, which costs at most $O(N^2)$, where N is the number of nodes in the network. Algorithm 2, lines 2-23 correspond an iterative process of optimization. The while loop runs at most $O(U(E_l)/e_r + ND_m) \leq O(N^2)$ times, where $U(E_l)$ is the upper bound of the network lifetime, which exists and is a constant, and D_m is the maximum degree of the network. It is because for each iteration, the algorithm either finds a type I eligible edge or a type II eligible edge or both. Finding type I eligible edge costs at most $O(U(E_l)/e_r)$ times as it improves the network lifetime by at least e_r each time. Finding type II eligible edge costs at most $O(ND_m)$ times as the bottleneck node only checks its candidate parents. Inside the iteration, the complexity of sorting is $O(D_m \log D_m)$, corresponding to Algorithm 2, line 3 and line 13; checking the neighbor nodes cost at most $O(D_m^{\;2})$, corresponding to lines 4-12 and lines 14-22. In summary, the total complexity is $O(N^2) \bullet (O(D_m \log D_m)+O(D_m^{\;2})) \leq O(N^4)$.

5. Simulation Results

In this section, we evaluate the performance of our algorithm by extensive simulations. We consider various network scenarios. In a typical scenario, we assume that 100 sensor nodes are uniformly distributed in a 100m×100m field. The sink (the 101th node) is located at the center of the field with its coordinate (50m, 50m). Each node has an initial energy of a randomly-generated value between 1J and 10J. The link PRR between two nodes are generated according to the model proposed in [14]. It is built on experimental measures of practical systems with respect to statistics of wireless channel. With the standard non-coherent FSK modulation and Manchester encoding, the PRR, $0 \leq p(d) \leq 1$, of a wireless link is expressible as:

$$p(d) = (1 - \frac{1}{2}\exp(-\frac{\gamma(d)}{2} \cdot \frac{1}{0.64}))^{8(2f-l)} \tag{9}$$

where d is the transmitter-receiver distance, $\gamma(d)$ is the signal-to-noise ratio (SNR), and f is the frame size including preamble l (2 bytes), payload and CRC. This model takes into account both distance-dependent path loss and log-normal shadowing in characterizing wireless links. For transmitting power P_t, the SNR, $\gamma(d)$, is expressible as:

$$\gamma(d) = P_t - PL(d) - P_n \quad \text{(dB)} \tag{10}$$

where P_t is set at -7 dB, the noise floor P_n is at -115 dB, and the path loss $PL(d)$ is modeled as:

$$PL(d) = PL(s_0) + 10n\log_{10}(d/d_0) + X_\sigma \quad \text{(dB)} \tag{11}$$

where n is the path loss exponent, d_0 is the reference distance (1 meter), and X_σ denotes the log-normal shadowing with zero mean and variance σ^2. In the coming simulations, we set n=3 and σ=7.

We assume that there is an edge between two nodes if and only if the PRR between them is larger than or equal to a threshold q_t. Unless elsewhere stated, q_t is set to 0.1. All simulation statistics are calculated over 1000 runs, with each run using a differently generated topology. The delay constraint D_c is set to different values in different scenarios. **Table 2** summarizes the commonly-used parameters in the simulations.

Table 2. Parameter settings in the simulation

Parameters	**Values**
Number of sensor nodes N	100~900
Area size A (m×m)	100×100, 200×200, 300×300
Sink position	The center of the field
Initial energy of each node $e(i)$ (J)	Random(1,10)
Packet length f (bit)	80
e_t (mJ)	600
e_r (mJ)	395
PRR threshold q_t	0.1

We compare the lifetime performance of the data gathering trees generated by five algorithms: 1) T_{LEG} (least ETX tree with greedy adjustment) generated by our proposed algorithm specified in Algorithm 2; 2) T_{LET} (least ETX tree) generated by Dijkstra's algorithm; 3) T_{SPS} (shortest path tree with semi-matching) generated by algorithm A_{SM1} specified in [3], without consideration of link quality, i.e., ETX is not included in the computation of expected node lifetime; 4) T_{SPM} (shortest path tree with modified semi-matching) also generated by algorithm A_{SM1}, yet taking link quality into consideration, i.e., expected node lifetime is computed by Eq. (3); 5) T_{RD} (random tree) generated by each node randomly choosing a node from its candidate parent set as its parent.

In the following evaluations, we demonstrate both numerical and simulation results. Numerical results are obtained by assuming that the link ETX is an exact measure of delay and retransmissions can guarantee the delivery, thus there will be no packet loss. However in practice, there will be packet loss if a maximum transmission count is reached. Furthermore, the actual transmission count is not in direct proportion to the link ETX, which implies that the link ETX is not an exact measure of delay. Therefore in simulation, we introduce a receiving deadline $D_r(T,i)$ and a transmission deadline $D_t(T,i)$ for each node i in tree T. Let $t_r(i)$ denote the epoch starting from the present round that node i receives all the packets from its children and $t_r(i,j)$ denote the epoch that node i receives the packet from node j. $D_r(T,i)$ and $D_t(T,i)$ are thus defined as follows:

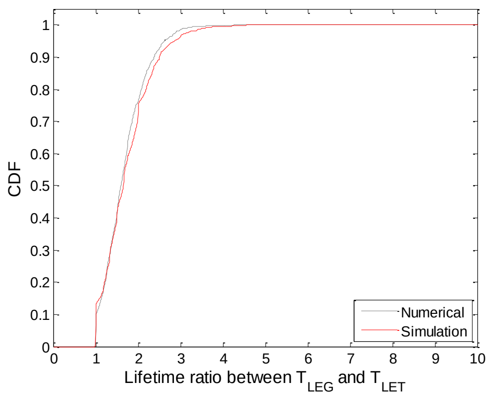
$$D_r(T,i) = \min(D_c - \lfloor E_t(T,i,0) \rfloor, t_r(i)) \tag{12}$$

$$D_t(T,i) = \min\left(D_r(T,i) + \lfloor 3E_t(i,p(T,i)) \rfloor, D_r(T,p(T,i)), t_r(p(T,i),i)\right) \tag{13}$$

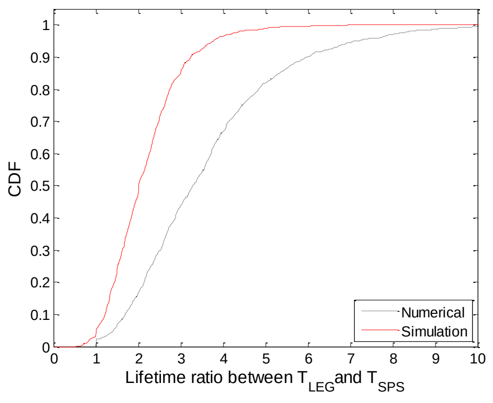
By Eq. (12), if node i has received packets from all of its children before time D_c - $E_t(T,i,0)$, it stops receiving, aggregates the packet and starts to transmit. Otherwise, it does aggregation at time D_c - $E_t(T,i,0)$ to guarantee that any node j in the upper hierarchy of the tree has at least $E_t(j,p(j))$ time for transmission. By Eq. (13), node i transmits at most $\lceil 3E_t(i,p(i)) \rceil$ times, as for any links with PRR>0.05, after 3/PRR transmission counts the probability of successful transmission will be larger than 95%. Node i stops the transmission when its parent stops receiving or when it receives ACK from its parent.

5.1 Lifetime Performance

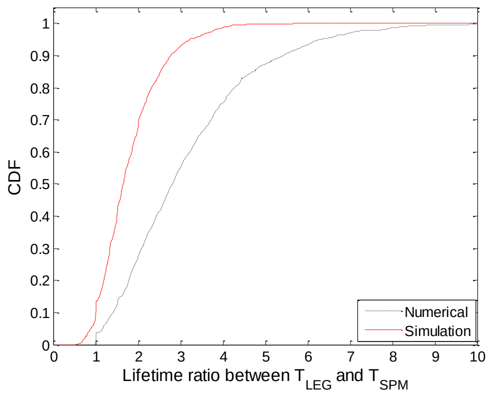
In the first set of simulations, we compare the lifetime performance of T_{LEG} generated by our algorithm with the other four trees T_{LET}, T_{SPS}, T_{SPM} and T_{RD} under different delay constraints. For each run, we compute the lifetime ratio between T_{LEG} and other trees. **Fig. 2** shows the cumulative distribution functions (CDF) of the lifetime gain over 1000 runs. As different trees yield different end-to-end delay, we cannot set a uniform delay constraint for all trees. Instead, for each comparison pair, we set the delay constraint to be the current end-to-end delay of respective trees to be compared. The lifetime performance of T_{LEG} significantly outperforms other schemes in all numerical situations and most of the simulated situations. It is worth noting that the simulation results deviate from the numerical ones, and moreover, in around 1% of the simulation situations, T_{LEG} cannot keep the ascendency. The reason is that the actual successful transmission count is randomly distributed, not in direct proportion to the link ETX. It is likely that a “worse” link actually need less transmission count than a “better” link. However, the tendencies of both results are accordant. Similar results are obtained with equal initial energy of the all the nodes. We believe that our algorithm will not be affected by the distribution of initial node energy. Due to limited space and similarity in results, we omit the lifetime performance with equal initial energy here,



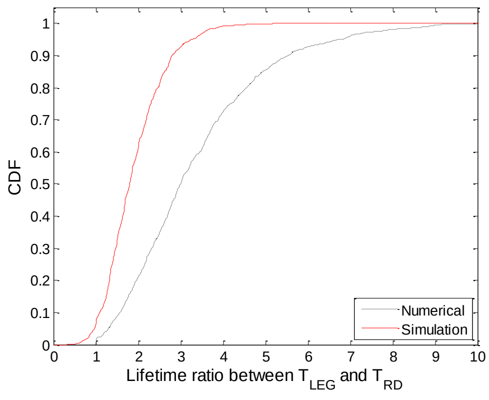
(a) D_c=D(T_{LET})



(b) D_c=D(T_{SPS})



(c) D_c=D(T_{SPM})

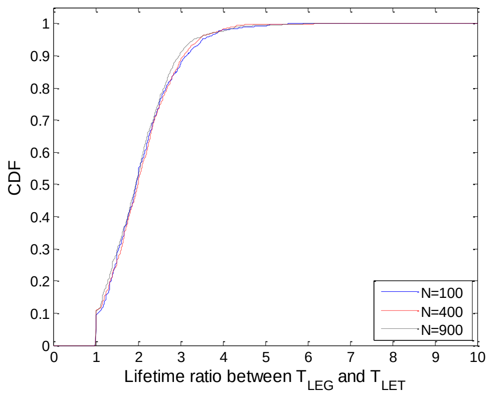


(d) D_c=D(T_{RD})

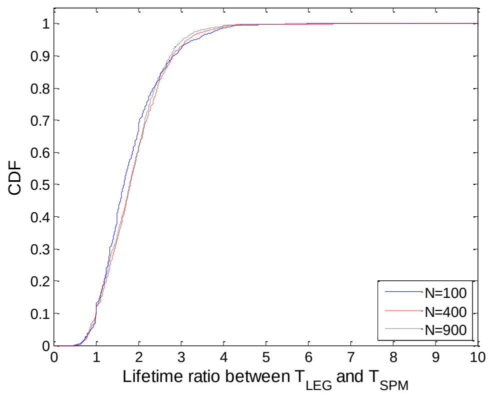
Fig. 2. CDF of the lifetime ratio between of T_{LEG} and other trees under respective delay constraints

5.2 Impact of Area Size

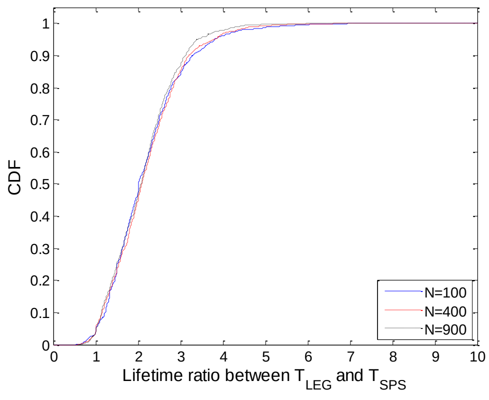
In the following, the CDF of lifetime gain under different area sizes are simulated and plotted in **Fig. 3**. The area size and the number of nodes are increased conformably to ensure an invariable node density. **Fig. 3** shows that T_{LEG} generated by our scheme has longer lifetime than other trees in most situations. Especially, as the area size increases, the dominance does not tail off, which verifies the fine scalability of our scheme. The reason that T_{SPS} and T_{SPM} can not obtain a superior lifetime performance is that they are basically constructed from a shortest path tree. In a network with unreliable links, a shortest path tends to be comprised of long links with unsatisfying link qualities. As a result, the shortest path incurs more transmissions and thus suffers from lifetime degradation. Sometimes, their lifetime performance is even worse than a random tree.



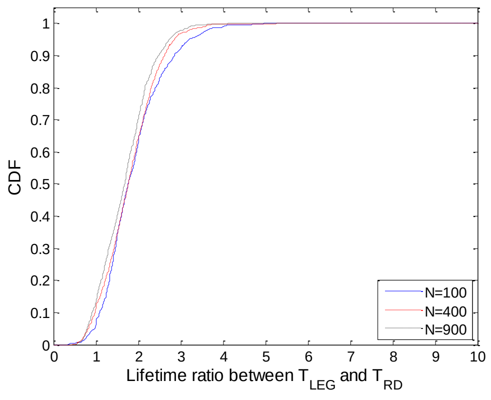
(a) D_c=D(T_{LET})



(b) D_c=D(T_{SPS})



(c) D_c=$D(T_{SPM})$



(d) D_c=$D(T_{RD})$

Fig. 3. Lifetime gain of our proposed approach over other approaches under different delay constraints and area sizes

5.3 Impact of Delay Constraint

The impact of delay constraints on the lifetime performance and data delivery rate is studied by simulation in this section. We plot the lifetime of T_{LEG} with the end-to-end delay constraint varying from $D(T_{LET})$ to $D(T_{LET})$ +10 (units). We also plot the numerical results. As shown in **Fig. 4**, with the increase of the delay constraint, the lifetime of T_{LEG} is becoming larger and finally converges to a constant value. The underlying reason is that when the delay constraint is stringent, there are few eligible edges satisfying the delay constraint. The adjustment for the bottleneck node is thus difficult to be made. However, as the delay constraint gets relaxed, there will be more eligible edges qualified to help the bottleneck node get released. In this way, the network lifetime can be improved. When the delay constraint is increased to some extent, the improvement space get smaller and smaller and finally converges to a constant state, which is close to the performance when there is no delay constraint. It is worth noting that there is a gap between numerical and simulation results. This is because in simulation, the maximum transmission count has been limited.

Fig. 5 shows the data delivery ratio of different trees. The data delivery ratio is defined as K/N, where K is the number of nodes whose packets are successfully delivered to the sink, and N is the total number of nodes. As aggregation is used in tree T, the packet loss of a parent node is in effect the loss of all the aggregated information from its subtree. For a given x, if the time for delivering the packet to the sink is larger than the delay constraint D_c=$D(T_{LET})$+x (units), the packet will be dropped and does not count in K. It is seen from **Fig. 5** that T_{LEG} and T_{LET} have almost 100% data delivery ratio. As T_{RD} and T_{SPS} does not consider link quality in tree construction, they experience a number of packet deadline misses. Therefore, the energy efficiency of T_{RD} and T_{SPS} will be degraded.

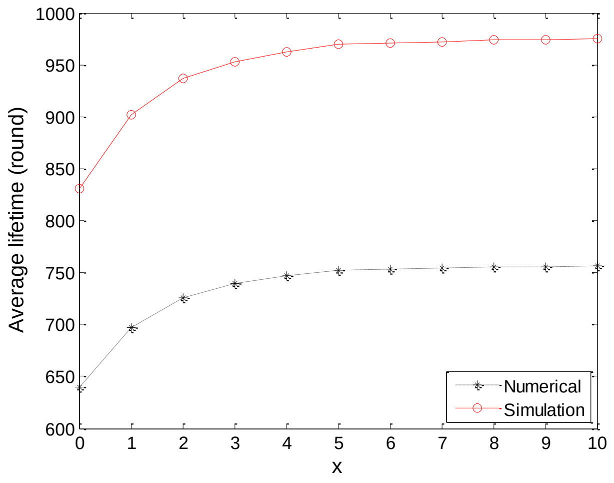


Fig. 4. The lifetime performance of T_{LEG} with the increase of delay constraint

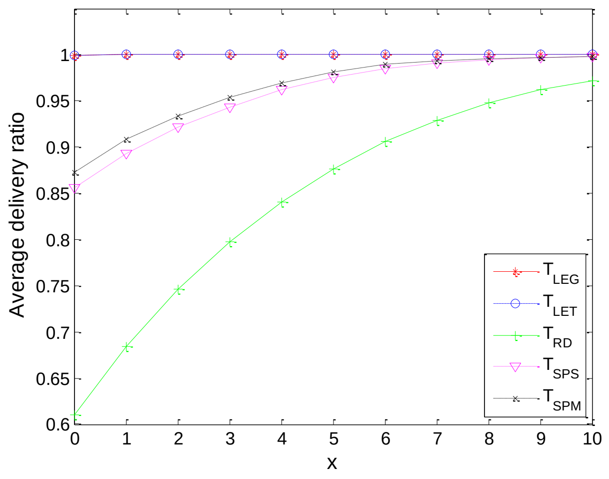


Fig. 5. The data delivery ratio of different trees with the increase of delay constraint

5.3 Impact of Node Density

In this subsection, we study the impact of node density on the lifetime performance. For comparison convenience, there is no delay constraint. We vary the number of nodes from 40 to 200 at a step of 20. **Fig. 6** shows the simulation results. The average lifetime of T_{LEG} is always several times larger than other trees. It is worth noting that, with the increase of node number, the lifetimes of T_{LET}, T_{SPM}, T_{SPS} and T_{RD} all take on a slight drop. On the contrary, the lifetime of T_{LEG} takes on a slight rise. The underlying causes are twofold: on one hand, increased number of nodes leads to an increase the total number of transmissions and the node degree, resulting in more energy consumption and energy distribution imbalance; on the other hand, the increase in node density can help improve the link quality of the network, resulting in less retransmission count. Regarding the twofold influence by the node number, we believe that, first, T_{LEG} is based on a least ETX tree, whose link quality benefits from the increase in node density; second, due to the recursive adjustment to the bottleneck node, T_{LEG} obtains a good energy balance throughout the network, which counteracts the energy consumption caused by augmented nodes. On the contrary, T_{SPS} is based on a shortest path tree, whose link quality cannot benefit from the increase in node density, while the node degree and the total transmission assignment are all the same increased. A slight drop in the lifetime is thus unavoidable.

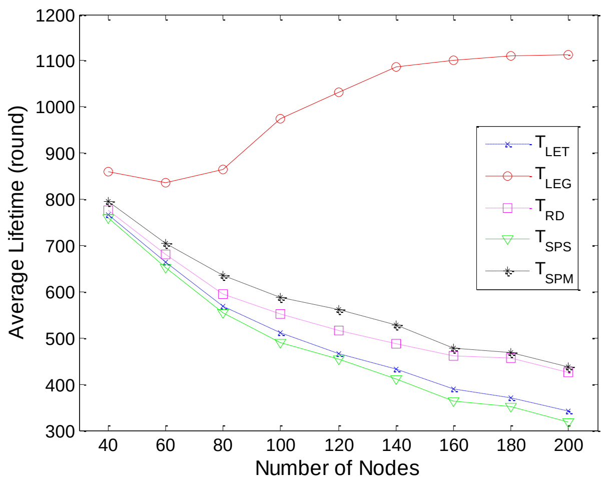


Fig. 6. The average network lifetime performance of different trees with the increase of node number

6. Conclusion

In this paper, we have studied the problem of the construction of a data gathering tree, which has maximum lifetime and satisfies end-to-end delay constraints, considering a lossy and unreliable wireless environment. This problem turns out to be NP-complete and difficult to be solved exactly. By investigating the inherent feature of the problem, we propose a greedy approximation algorithm which can terminate in polynomial time. Our algorithm starts from an arbitrary least ETX tree, and iteratively adjusts the hierarchy of the tree to reduce the load on bottleneck nodes. Meanwhile, the adjustment is only allowed with the guarantee of end-to-end delay constraint. Extensive simulations are carried out to verify this approach. Simulation results show that our algorithm provides longer lifetime in various situations compared to exiting popular data gathering schemes.

Currently, the definition of network lifetime used in this paper may be too stringent. In practice, the network should tolerate the death of a small number of nodes as long as the quality of aggregated/fused information is satisfying. Tradeoffs between the information quality and the network lifetime are worth investigating in the future. Furthermore, the implementation of our algorithm in this work is confined to numerical simulations. An implementation on real sensor network platforms is our ongoing work.

References

[1] S. Hariharan and N. B. Shroff, “On optimal energy efficient convergecasting in unreliable sensor networks with applications to target tracking,” in *Proc. of ACM MobiHoc*, pp. 1-10, May 2011. Article (CrossRef Link).

[2] S. He, J. Chen, D. K.Y. Yau and Y. Sun, Cross-layer Optimization of Correlated Data Gathering in Wireless Sensor Networks, *IEEE Trans. on Mobile Computing*, vol. 11, no. 11, pp. 1678-1691, 2012. Article (CrossRef Link).

[3] 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 *Proc. of IEEE INFOCOM*, pp. 1566-1574, April 15-17, 2008. Article (CrossRef Link).

[4] D. Luo, X. Zhu, X. Wu, and G. Chen, “Maximizing lifetime for the shortest path aggregation tree in wireless sensor networks,” in *Proc. of IEEE INFOCOM*, pp. 1566-1574, April 15-17, 2011. Article (CrossRef Link).

[5] X. Wang, X. Wang, G. Xing, and Y. Yao, “Exploiting overlapping channels for minimum power configuration in real-time sensor networks,” In *Proc. of the 7th European conference on Wireless Sensor Networks* (EWSN'10), pp. 97-113, February 17-19, 2010. Article (CrossRef Link).

[6] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher, “A spatiotemporal communication protocol for wireless sensor networks,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 16, no. 10, pp. 995-1006, 2005. Article (CrossRef Link).

[7] E. Felemban, C. G. Lee, and E. Ekici, “MMSPEED: Multipath multispeed protocol for QoS guarantee of reliability and timeliness in wireless sensor network,” *IEEE Trans. on Mobile Computing*, vol. 5, no. 6, pp. 738-754, 2006. Article (CrossRef Link).

[8] Y. Li, C. S. Chen, Y. Q. Song, Z. Wang, and Y. Sun, “Enhancing realtime delivery in wireless sensor networks with two-hop information,” *IEEE Trans. on Industrial Informatics*, vol. 5, no. 2, pp. 113-122, 2009. Article (CrossRef Link).

[9] T. L. Lim and G. Mohan, “Energy aware geographical routing and topology control to improve network lifetime in wireless sensor networks,” in *Proc. of 2nd International Conference on Broadband Networks*, pp. 771-773, 2005. Article (CrossRef Link).

[10] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy efficient communication protocol for wireless microsensor networks,” in *Proc. of 33rd Annual Hawaii International Conference on System Sciences*, pp. 3005-3014, 2000. Article (CrossRef Link).

[11] O. Younis and S. Fahmy, “HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks,” *IEEE Trans. Mobile Computing*, vol. 3, no. 3, pp. 366-379, 2004.

[12] S. Lindsey and C. S. Raghavendra, “PEGASIS: Power-efficient gathering in sensor information systems,” in *Proc. of IEEE Aerospace Conference*, pp. 1125-1130, 2002. Article (CrossRef Link).

[13] H. Tan and I. K¨orpeoglu, “Power efficient data gathering and aggregation in wireless sensor networks,” *ACM SIGMOD Record*, vol. 32, no. 4, pp. 66-71, 2003. Article (CrossRef Link).

[14] M. Zuniga and B. Krishnamachari, “Analyzing the transitional region in low power wireless links,” in *Proc. of IEEE SECON*, pp. 517-526, 2004. Article (CrossRef Link).

[15] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, “Collection tree protocol,” in *Proc. of ACM SenSys*, pp. 1-14, 2009. Article (CrossRef Link).

[16] C. J. M. Liang, J. Liu, L. Luo, A. Terzis, and F. Zhao, “Racnet: a highfidelity data center sensing network,” in *Proc. of ACM SenSys*, pp. 15-28, 2009. Article (CrossRef Link).

[17] J. Chen, W. Xu, S. He, Y. Sun, P. Thulasiramanz and X. Shen. Utility-Based Asynchronous Flow Control Algorithm for Wireless Sensor Networks. *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 7, pp. 1116-1126, 2010. Article (CrossRef Link).

[18] L. Sang, A. Arora and H. Zhang, "On exploiting asymmetric wireless links via one-way estimation," in *Proc. of ACM MobiHoc*, pp. 11-21, September 9-14, 2007. Article (CrossRef Link).

[19] IEEE Std 802.15.4, “Part 15.4: Wireless medium access (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs),” *IEEE-SA Standards Board*, September 2006. Article (CrossRef Link).

[20] I Dietrich, F Dressler, “On the lifetime of wireless sensor networks”, *ACM Trans. on Sensor*

Networks, vol. 5, no. 1, Article 5, February 2009. Article (CrossRef Link).
[21] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269-271, 1959. Article (CrossRef Link)

Yanjun Li received the B.S. and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 2004 and 2009, respectively, and another Ph.D. degree from Nancy University, Villers-les-Nancy, France, in 2010. She is currently an Associate Professor in School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China. Her research area includes ad hoc network, sensor network, internet of things and social network. She is a member of IEEE and has published more than 30 referred technical papers in proceedings and journals

Yueyun Shen received the B.S. degree from Information Management and Information System, Zhejiang Agriculture and Forestry University, Lin'an, Hangzhou, China, in 2009. She is currently working toward the Master degree in School of Computer Science and Technology, Zhejiang University of Technology, China. Her major research interests are cooperative routing, wireless sensor network and distributed system.

Kaikai Chi received the B.S. and M.S. degrees from Xidian University, Xi'an, China, in 2 and 2005, respectively, and the Ph.D. degree from Tohoku University, Sendai, Japan, in 2(He is currently an Associate Professor in School of Computer Science and Technolo Zhejiang University of Technology, Hangzhou, China. His current research focuses on wire ad hoc network and wireless sensor network. He was the recipient of the Best Paper Awar the IEEE Wireless Communications and Networking Conference in 2008. He is a membe IEEE and has published more than 20 referred technical papers in proceedings and jour including IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions Vehicular Technology, etc.