Mobile Networks and Applications 10, 853–864, 2005

DOI: 10.1007/s11036-005-4443-7

Maximizing Lifetime for Data Aggregation in Wireless Sensor Networks

YUAN XUE, YI CUI and KLARA NAHRSTEDT
Department of Electrical Engineering and Computer Science, Vanderbilt University, VU Station B 351824, Nashville, TN 37235, USA

Published online: 24 October 2005

Abstract. This paper studies energy efficient routing for data aggregation in wireless sensor networks. Our goal is to maximize the lifetime of the network, given the energy constraint on each sensor node. Using linear programming (LP) formulation, we model this problem as a multicommodity flow problem, where a commodity represents the data generated from a sensor node and delivered to a base station. A fast approximate algorithm is presented, which is able to compute $(1-\epsilon)$-approximation to the optimal lifetime for any $\epsilon > 0$. Then along this baseline, we further study several advanced topics. First, we design an algorithm, which utilizes the unique characteristic of data aggregation, and is proved to reduce the running time of the fastest existing algorithm by a factor of K, K being the number of commodities. Second, we extend our algorithm to accommodate the same problem in the setting of multiple base stations, and study its impact on network lifetime improvement. All algorithms are evaluated through both solid theoretical analysis and extensive simulation results.

Keywords: sensor network, multicommodity flow problem, linear programming

1. Introduction

The convergence of micro-eletro-mechanical system technology, wireless communication and digital electronics leads to the emergence of wireless networks of sensor devices [1], which are capable of sensing, data processing, and communicating. Sensor networks can be readily deployed in diverse environments to collect and process useful information in a autonomous manner. Thus, they have a wide range of applications in the areas of health care, military, and disaster detection.

One of the basic operations of a sensor network is data aggregation [17,19]. During the data aggregation, sensed data is gathered from different sensors (data source), combined at intermediate nodes, and eventually transmitted to the base station (data sink) for further processing. One of the most important challenges in designing an efficient data aggregation scheme is the energy constraint—sensor nodes carry limited, irreplaceable, power supply. As radio communication consumes a large fraction of this supply, it is critical to design *energy-efficient* routing algorithms for data aggregation in wireless sensor networks. In the existing works, the problem of designing energy-efficient routing algorithms has been extensively studied in both general multihop wireless networks [5,6,18,23,24], and the particular backdrop of sensor networks [4,7–9,21]. These energy-efficient routing algorithms can achieve the goal of either minimizing energy consumption [13,22,24,26–29], or maximizing the network lifetime [5,6,8,20,23].

To achieve the goal of minimum energy routing, the typical approach [13,22] is to use a shortest path algorithm in which the edge cost is the power consumed to transmit a packet between two nodes of this edge. Though effectively reducing the energy consumption rate, this approach can cause unbalanced consumption distribution, i.e., the nodes on the minimum-energy path are quickly drained of energy, causing network partition or malfunctioning. Therefore, we consider the problem of maximizing network lifetime, which is a more critical goal in the context of wireless sensor network, i.e., to maximally prolong the duration in which the entire network properly functions. Although this problem has been studied in existing literatures [4,20], many of its crucial aspects remain uninvestigated. Among many of them, we are interested with the following:

- Given the topological layout and energy reserve of a sensor network, how to obtain the upper bound of its lifetime, and the optimal routing schedule to achieve this bound? Furthermore, what unique characteristics of data aggregation could we utilize to come up with a more efficient solution than existing approaches?
- Given the answer to the first problem, how could we extend our solution to accommodate the same problem in more advanced settings, e.g., how to quantify the performance gain if multiple data sinks are distributed within one network?

To answer these questions, we model the problem of *maximizing network lifetime* as a *concurrent multicommodity flow* problem. In this formulation, a commodity represents the sensed data from a sensor delivered to a base station with certain generating rate. Given the sending rate of each commodity and energy consumption rate to send unit flow from one node to another within the network, the objective is to maximize the lifetime of the network with the constraint of the energy reserve on each node. Since multicommodity flow is one of the classical linear programming (LP) problems, we can address our problem using standard LP solving techniques. Here, we adopt the Garg-Konemann algorithm [12], a fast approximate algorithm which computes $(1-\epsilon)$-approximation to the

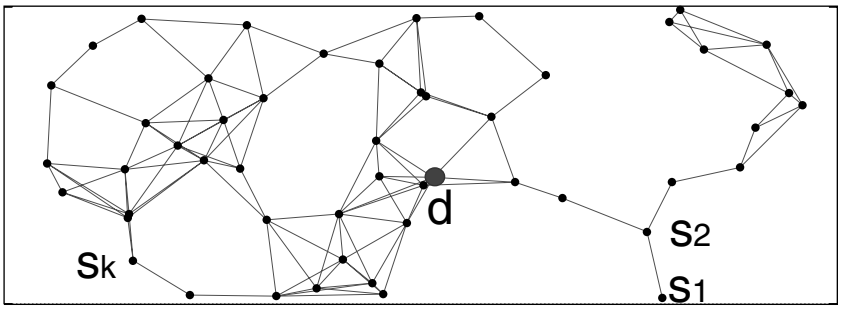


Figure 1. A sensor network.

optimal objective value for any $\epsilon > 0$. In this algorithm, a weight is assigned to each node as a function of its current load (amount of data routed via it). Based on these weights, the algorithm finds the shortest path for each commodity, such that the weighted sum of energy consumption rates of all nodes along this path is minimum, then adds certain amount of data on this route. Such a procedure is repeated in iterations until certain stopping condition is met.

We then focus on the unique characteristic of the data aggregation problem, i.e., all data sources share the same destination, to further improve the algorithm performance. Note that in the Garg-Konemann algorithm, a shortest path is calculated for each source and destination pair in each iteration. In the case of data aggregation, if a shortest path tree is calculated between the destination and all sources in each iteration, the algorithm's running time can be greatly improved. Following this intuition, we formally present the concept of *aggregation tree*, which is a unification of unicast routes from all sources to the common destination. Based on this concept, we re-formulate the problem of maximizing network lifetime for data aggregation and extend the traditional path-based algorithm to a tree-based algorithm. This tree-based algorithm calculates the data routes per aggregation tree in each iteration. We prove that this algorithm reduces the running time of the fastest existing algorithm by a factor of K, K being the number of commodities. Finally, we extend the algorithm for the case of multiple data sinks and show that its running time property still holds. Using this algorithm, we further study the impact of the data sink numbers on the network lifetime via simulation.

The main contributions of this paper are as follows. First, it extends the framework of multicommodity flow problem to address the unique characteristic of data aggregation in network lifetime maximization. The proposed tree-based algorithm is shown to greatly reduce the running time, compared to existing algorithms [5,6], and achieve better scalability in terms of network size than existing approach [8]. Second, based on this improved algorithm, this paper studies the network lifetime maximization problem when multiple data sinks exist in data aggregation and evaluate the impact of number of sinks on the network lifetime. To the best of our knowledge, this paper is the first work that performs such studies using a formalized method.

The rest of this paper is organized as follows. We model the network and formulate the general network lifetime maximization problem in Section 2. Section 3 presents the aggregation-tree-based formulation, algorithm and analysis. Section 4 presents the extended algorithm for the multiple-sink problem. Section 5 shows the performance study, Section 6 presents the related work, and Section 7 concludes the paper.

2. Model and problem formulation

2.1. Network model

Consider a wireless sensor network with K sensors, denoted as $s_1, s_2, \ldots, s_K$ and a base station d as shown in figure 1. The locations of the sensors and the base station are fixed and known a priori. We model such a network using a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where the node set $\mathcal{N} = \{d, s_1, s_2, \ldots, s_K\}$. Each sensor node $s_i, (i = 1, \ldots, K)$ is associated with E_i, representing its energy reserve. We assume that the base station d has infinite power supply. Each edge $(n, n') \in \mathcal{L}$[1] is associated with a cost $c^t(n, n')$, representing the energy required to transmit one unit of data from node n to n', and a cost c^r, representing the energy required to receive one unit of data at node n. The first order radio model shows that costs $c^t(n, n')$, c^r can be represented as follows.

$$c^t(n, n') = \alpha + \beta \cdot (\delta_{nn'})^m \tag{1}$$

$$c^r = \alpha \tag{2}$$

where α is a distance-independent constant that represents the energy consumption to run the transmitter or receiver circuitry, β is a distance-dependent term that represents the transmitter amplifier, and $\delta_{nn'}$ is the distance between the antennas of node n and n'. The exponent m is determined from field measurements, which is typically a constant between 2 and 4.

In this paper, we focus on the data aggregation problem. In data aggregation, each sensor is a *data source* which produces some information as it monitors its vicinity. Such information

[1] Here we use n, n' to represent the node which can be either a sensor or the base station.

Table 1
Notation in Section 2.

Notation	Description
s_i $(i = 1, \ldots, K)$	Sensor nodes, data source
d	Base station node, data sink
$\mathcal{G} = (\mathcal{N}, \mathcal{L})$	Network with node set $\mathcal{N}$ and edge set $\mathcal{L}$
$n \in \mathcal{N}$	Nodes in the network
$(n, n') \in \mathcal{L}$	Wireless link from n to n'
E_i	Energy reserve of sensor node s_i $(i = 1, \ldots, K)$
$c^t(n, n')$	Energy cost of sending one unit of data via (n, n')
c^r	Energy cost of receiving one unit of data
$D_i (i = 1, \ldots, K)$	Demand (rate) of commodity i
$\mathcal{P}_i = \{P_j^i\} (i = 1, \ldots, K)$	Set of paths from s_i to d
$f(P_j^i)$	Amount of commodity sent via ${P_j}^i$
$c_k(P_j^i)$	Energy cost of sensor node s_k for unit data along ${P_j}^i$
w_k	Weight assigned to sensor node s_k
T	System lifetime

will be gathered and sent to the base station, which is the *data sink*, directly or via the relays of other sensor nodes. Here we assume that at relay nodes data are assembled, without further processing (such as compression). The information delivery from each data source $s_i, (i = 1, \ldots, K)$ to the sink d forms *commodity* i. We further assume that the rate of commodity i is D_i, which is called the *demand* for this commodity. In addition, we denote the set of paths exist between s_i and d as $\mathcal{P}_i = \{P_j^i\}$. Each commodity can be arbitrarily split and sent along several paths in parallel. We use $f(P_j^i)$ to denote the total amount of the commodity i sent along the path P_j^i. We further associate a cost $c_k(P_j^i)$ with sensor node $s_k, (k = 1, \ldots, K)$, which represents the per unit commodity power consumption incurred at node s_k by commodity i that is delivered along path P_j^i. Based on equation (1), we have that

$$c_k\left(P_j^i\right) = \begin{cases} c^t(k, n') + c^r & \text{if } s_k \text{ is a relay node for} P_j^i, \text{ and} \\ & \text{link}(k, n') \text{ lies on the path } P_j^i \\ c^t(k, n') & \text{if } s_k \text{ is the source node for} P_j^i, \text{ and} \\ & \text{link } (k, n') \text{ lies on the path } P_j^i \end{cases} \tag{3}$$

Table 1 lists the notations that are introduced in this section.

2.2. Problem formulation: Concurrent multicommodity flow

Now we formulate the *network lifetime maximization* problem as a *concurrent multicommodity flow* problem. Let us denote the network lifetime as T, which is the time the network keeps functioning until one node drains its energy. The objective of this problem is to maximize T, subject to the flow conservation and energy constraints. Using the linear programming (LP) formulation, we have

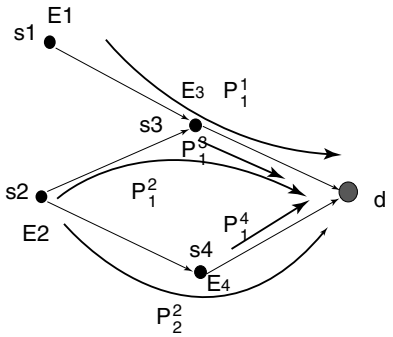


Figure 2. Concurrent multicommdity flow problem. In the figure, the flow constraints are $f(P_1^1) \geq T \cdot D_1$, $f(P_1^2) + f(P_2^2) \geq T \cdot D_2$, $f(P_1^3) \geq T \cdot D_3$, and $f(P_1^4) \geq T \cdot D_4$. And the energy constraints are $c_1(P_1^1) \cdot f(P_1^1) \leq E_1$, $c_2(P_1^2) \cdot f(P_1^2) + c_2(P_2^2) \cdot f(P_2^2) \leq E_2$, $c_3(P_1^1) \cdot f(P_1^1) + c_3(P_1^2) \cdot f(P_1^2) + c_3(P_1^3) \cdot f(P_1^3) \leq E_3$, and $c_4(P_2^2) \cdot f(P_2^2) + c_4(P_1^4) \cdot f(P_1^4) \leq E_4$.

$$\mathbf{P}: \text{ maximize } T \tag{4}$$

$$\text{subject to} \sum_{j=1}^{|\mathcal{P}_i|} f\left(P_j^i\right) \geq T \cdot D_i, \quad i = 1, \ldots, K \tag{5}$$

$$\sum_{i=1}^{K} \sum_{j=1}^{|\mathcal{P}_i|} c_k\left(P_j^i\right) \cdot f\left(P_j^i\right) \leq E_k, \quad k = 1, \ldots, K \tag{6}$$

$$T \geq 0, f\left(P_j^i\right) \geq 0, \forall i, \forall j \tag{7}$$

In this formulation, equation (5) shows the constraint of concurrent flow, where the total amount of commodity from a sensor node should be no less than its demand over the network lifetime. Equation (6) reflects the energy constraint, where the total amount of energy consumption on sensor s_k for all commodities $i = 1, \ldots, K$ should be no larger than its energy reserve. Such a problem formulation is illustrated in figure 2.

There are several flavors of solving techniques to a LP problem. We are interested to find a fully polynomial time approximation scheme (FPTAS) to this problem. FPTAS is a family of algorithms that find an ϵ-approximate solution, which returns a result at least $(1 - \epsilon)$ times the optimal value, for any error parameter $\epsilon > 0$. Its running time is polynomial in the size of the network ($|\mathcal{N}|$ and $|\mathcal{L}|$), the number of commodities (K), and $1/\epsilon$. In this section, we propose a FPTAS to P based on the scheme proposed by Garg and Konemann [12], which was later improved by Fleischer [10]. By LP duality theory [14], we first formulate the dual problem of **P** as

$$\mathbf{D(P)}: \text{ minimize} \sum_{k=1}^{K} E_k \cdot w_k \tag{8}$$

$$\text{subject to} \sum_{k=1}^{K} c_k(P_j^i) \cdot w_k \geq l_i, P_j^i \in \mathcal{P}_i,$$

$$i = 1, \ldots, K, \forall j$$

$$\sum_{i=1}^{K} l_i \cdot D_i \geq 1$$

Table 2
Algorithm for maximum concurrent flow problem.

MaxConcurrentFlow

1 $w_k \leftarrow \beta/E_k, k = 1, \ldots, K$
2 $\forall i, j, k = 1, \ldots, K, c_k(P_j^i) \leftarrow w_k \cdot c_k(P_j^i)$
3 $f(P_j^i) \leftarrow 0, P_j^i \in \mathcal{P}_i, i = 1, \ldots, K$
4 **while** $\sum_{k=1}^{K} E_k \cdot w_k < 1$
5 **for** $i = 1$ *to* K **do**
6 $D_i' \leftarrow D_i$
7 **while** $\sum_{k=1}^{K} E_k \cdot w_k < 1$ **and** $D_i' > 0$
8 $P \leftarrow$ shortest path in $\mathcal{P}_i$ with the cost of link (n, n') as $c^t(n, n') + c^r$
9 $e \leftarrow \min\{D_i', \min_{k \in P} \frac{E_k}{c_k(P)}\}$
10 $D_i' \leftarrow D_i' - e$
11 $f(P) \leftarrow f(P) + e$
12 **for** $\forall s_k \in P$ **do**
13 $w_k \leftarrow w_k(1 + \epsilon \frac{c_k(P)e}{E_k})$
14 $\forall s_k \in P, c_k(P) \leftarrow c_k(P)(1 + \epsilon \frac{c_k(P)e}{E_k})$
15 **end for**
16 **end while**
17 **end for**
18 **end while**
19 Scale $f(P_j^i)$ *by* $\log_{1+\epsilon} 1/\beta, \forall i, j$

$$w_k \geq 0, l_i \geq 0, i, k = 1 \ldots, K$$

D(P) corresponds to the problem of assigning weight w_k to each sensor node s_k and weight l_i to each commodity i, such that for commodity i, the weighted cost of any path $P_j^i \in \mathcal{P}_i$ (i.e., the sum of each node k's energy cost scaled by its weight w_k) is at least l_i, and the weighted sum of l_i by D_i over all commodities is at least 1. By LP duality theory, the minimum of **D(P)** is the maximum of **P**. Here, w_k represents the marginal cost of using an additional unit of s_k's energy reserve, and l_i represents the marginal cost of not satisfying another unit of the demand D_i.

Based on Garg and Konemann [21], we present an algorithm for this problem, henceforth referred to as **MaxConcurrentFlow**, in Table 2. Line (1)–(3) initialize the algorithm with $w_k = \beta/E_k$ for each sensor node s_k, scale $c_k(P_j^i)$ by w_k, and set $f(P_j^i) = 0$ for each i, j, k. Then the algorithm proceeds in phases. In each phase, there are K iterations. In iteration i, the objective is to route D_i units of commodity for the commodity i. This is done in steps. In one step, a shortest path P is first computed using $c^t(n, n') + c^r$ as the cost of link (n, n') (Line 8). Then e units of commodity, which is constrained by its bottleneck energy reserve, is sent along P (Line 9). If e already exceeds the remaining demand $D_i{}'$, we only send $D_i{}'$ along P (Line 10). Finally, for each node s_k on path P, we augment w_k and $c_k(P)$ by the factor $(1 + \epsilon \frac{c_k(P)e}{E_k})$ (Line 13 and 14). The entire procedure stops when the objective function value is at least one: $\sum_{k=1}^{K} E_k \cdot w_k \geq 1$. Finally, scaling the final flow by $\log_{1+\epsilon} 1/\beta$ (Line 19) yields a feasible solution to the problem. Note that our algorithm routes the flow for each commodity in proportion to its demand, namely

$$\frac{\sum_{j=1}^{|\mathcal{P}_1|} f(P_j^1)}{D_1} = \frac{\sum_{j=1}^{|\mathcal{P}_2|} f(P_j^2)}{D_2} = \ldots = \frac{\sum_{j=1}^{|\mathcal{P}_K|} f(P_j^K)}{D_K} = T^*$$

Furthermore, the maximum lifetime T^* is the ratio of any commodity's final flow and its demand. Following the same way as Garg and Konemann, we have the following result.

Proposition 1. When $\beta = (\frac{1-\epsilon}{K})^{1/\epsilon}$, **MaxConcurrentFlow** computes a $(1 - 3\epsilon)$-approximation to the maximum concurrent flow problem, with running time $O(\frac{K^2 \log K}{\epsilon^2} \cdot T_{sp})$.

T_{sp} is the running time of the shortest path algorithm. For example, $T_{sp} = O(|L| + K \log K)$ under Dijkstra's shortest path algorithm.

3. Aggregation-tree-based algorithm

Using the basic formulation and algorithm presented in Section 2, we now further explore the unique property of data aggregation for more efficient solutions. Note that in each iteration of the **MaxConcurrentFlow** algorithm, a shortest path is found for *one* commodity, the cost and weight of each node s_k $(k = 1, \ldots, K)$ are then updated. The algorithm stops when the value of the objective function $\sum_{k=1}^{K} E_k \cdot w_k \geq 1$.

Intuitively, to speed up the running time of this algorithm, shortest paths can be found for more commodities in each iteration, so that more traffic can be routed, and the growth of each node's weight can accelerate, i.e., the number of rounds to reach the threshold 1 can be reduced. In our problem, since all data sources share a common destination, we can find all their shortest paths in the one-to-many way, i.e., a shortest path tree rooted at the data sink d. It is well known that Dijkstra's algorithms for single-pair shortest path problem and shortest path tree problem have the same performance bound $O(|\mathcal{L}| + K \log K)$. Though intuitive, proving the correctness of this idea and deriving its property is non-trivial. In the rest of this section, we formally present the algorithm and analyze its property.

3.1. Problem reformulation: Aggregation-tree-based flow problem

Following above intuition, we propose the concept of *aggregation tree*, which will be used to reformulate the problem and give formal proof of above intuitive idea. Each aggregation tree has the sink d as its root, and spans all data sources s_i $(i = 1, \ldots, K)$. We use $\mathcal{S} = \{S_j\}$ to denote the set of aggregation trees. Each tree S_j can be decomposed into K paths $P_j^i (i = 1, \ldots, K)$ for each commodity. The flows on each path are in proportion to their demands, namely

$$\frac{f(P_j^1)}{D_1} = \frac{f(P_j^2)}{D_2} = \cdots = \frac{f(P_j^K)}{D_K} \tag{9}$$

The flow rate of S_j is the aggregated rate from all sources s_i $(i = 1, \ldots, K)$ to d, i.e., $f(S_j) = \sum_{i=1}^{K} f(P_j^i)$. The problem

Table 3
Notations in Section 3.

Notation	Description
$\mathcal{S} = \{S_j\}$	Set of aggregation trees from all sources s_i $(i = 1, \ldots, K)$ to d
$f(S_j)$	Amount of data flow sent via S_j
$c_k(S_j)$	Energy cost of node s_k if one unit of data is sent via S_j

Table 4
Transferring solution of **P** to solution of **S**.

Transfer
1 $j \leftarrow 1\ \Gamma \leftarrow \phi$
2 **while** $T^* > 0$
3 **for** $i = 1$ **to** K **do** $P^i_{max} \leftarrow$ **ExtractMax**(Ω_i)
4 $S_j \leftarrow \bigcup_{i=1}^{K} P^i_{max}$
5 $f(S_j) \leftarrow \min_{i=1}^{K}\{\frac{f(P^i_{max})}{D_i}\} \cdot \sum_{i=1}^{K} D_i$
6 $\Gamma \leftarrow \Gamma \cup S_j$
7 $T^* \leftarrow \frac{f(S_j)}{\sum_{i=1}^{K} D_i}$
8 **for** $i = 1$ **to** K **do**
9 $f(P^i_{max}) \leftarrow f(P^i_{max}) - \frac{f(S_j)}{\sum_{i=1}^{K} D_i}$
10 **if** $f(P^i_{max}) > 0$ **then** $\Omega_i \leftarrow \Omega_i \cup P^i_{max}$
11 **end for**
12 $j \leftarrow j + 1$
13 **end while**

P is then reformulated as

$$\mathbf{S}: \text{maximize} \sum_{j=1}^{|\mathcal{S}|} f(S_j) \tag{10}$$

$$\text{subject to} \sum_{j=1}^{|\mathcal{S}|} c_k(S_j) \cdot f(S_j) \le E_k, \quad k = 1, \ldots, K$$

$$f(S_j) \ge 0, \forall j$$

Here $c_k(S_j)$ is defined as follows.

$$c_k(S_j) \triangleq \frac{\sum_{i=1}^{K} D_i \cdot c_k(P^i_j)}{\sum_{i=1}^{K} D_i}$$

Table 3 lists the new notations introduced in this section. To show that **P** and **S** are equivalent regarding the problem of data aggregation, we need to show the one-to-one mapping relationship of **P** and **S**'s solution spaces. We first show a simple algorithm (Table 4) transferring a solution of **P** to a solution of **S**. In the algorithm, T^* is the lifetime calculated by the algorithm **MaxConcurrentFlow**. $\Omega_i (i = 1, \ldots, K)$ includes paths found for commodity i. Ω_i is sorted based on the amount of flow on its paths. The algorithm transfers paths in Ω_i into aggregation trees, which are collected in Γ. Each time a tree is constructed, at least one path's flow is totally moved to Γ. Thus the number of rounds is at most $\sum_{i=1}^{K} |\Omega_i|$. Also from the algorithm we can see, each time a new tree S_j is constructed, T^* is decreased by $\frac{f(S_j)}{\sum_{i=1}^{K} D_i}$. Thus, the final result of the solution to **S** is in proportion to T^*, i.e., $T^* = \frac{\sum_{j=1}^{|\mathcal{S}|} f(S_j)}{\sum_{i=1}^{K} D_i}$.

Table 5
Algorithm for the aggregation tree problem.

AggregationTree
1 $k = 1, \ldots, K, w_k \leftarrow \beta / E_k$
2 $\forall i, j, k = 1, \ldots, K, c_k(S_j) \leftarrow w_k \cdot c_k(S^i_j)$
3 $f(S_j) \leftarrow 0, S_j \in \mathcal{S}$
4 **while** $\sum_{k=1}^{K} E_k \cdot w_k < 1$
5 $S \leftarrow$ shortest path tree rooted at D in $\mathcal{S}$ with the cost of link (n, n')as $c^t(n, n') + c^r$
6 $e \leftarrow \min_{s_k \in S} \frac{E_k}{c_k(S)}$
7 $f(S) \leftarrow f(S) + e$
8 **for** $\forall s_k \in S$ **do**
9 $w_k \leftarrow w_k(1 + \epsilon \frac{c_k(S)e}{E_k})$
10 $\forall s_k \in S, c_k(S) \leftarrow c_k(S)(1 + \epsilon \frac{c_k(S)e}{E_k})$
11 **end for**
12 **end while**
13 Scale $f(S_j)$ by $\log_{1+\epsilon} 1/\beta, \forall j$

Transferring a solution of **S** to a solution of **P** is more straightforward. Suppose Γ collects all aggregation trees, we just need to decouple each tree in Γ into K paths for each commodity, and feed them into $\Omega_i (i = 1, \ldots, K)$ separately.

Now we proceed to study the solution to **S**. We first formulate the dual problem of **S** as

$$\mathbf{D(S)}: \text{minimize} \sum_{k=1}^{K} E_k \cdot w_k$$

$$\text{subject to} \sum_{k=1}^{K} c_k(S_j) \cdot w_k \ge 1, \quad j = 1, \ldots, |\mathcal{S}|$$

$$w_k \ge 0, k = 1, \ldots, K \tag{11}$$

D(S) corresponds to the problem of assigning weight w_k to each node $s_k, (k = 1, \ldots, K)$, such that the weighted cost (the sum of each node's energy cost scaled by its weight) of any tree in $\mathcal{S}$ is at least 1. By LP duality theory, the minimum of **D(S)** is the maximum of **S**. Here, w_k represents the marginal cost of using an additional unit of s_k's energy reserve.

3.2. Algorithm

Based on **D(S)**, the algorithm **AggregationTree** is shown in Table 5. Line (1)–(3) initialize the algorithm with $w_k = \beta / E_k$ for each sensor node s_k, scale $c_k(P^i_j)$ by w_k, and set $f(S_j) = 0$ for each i, k and each tree $S_j \in \mathcal{S}$. In each iteration, a shortest path tree S is computed using $c^t(n, n') + c^r$ as the cost of link (n, n'). S is rooted at the sink d, and spans all data sources s_i $(i = 1, \ldots, K)$ (Line 5). We then send e units of commodity along S, which is the bottleneck energy constraint of $S: e = \min_{s_k \in S} \frac{E_k}{c_k(S)}$ (Line 6). Finally, for each node s_k on S, we augment w_k and $c_k(S)$ by the factor $1 + \epsilon \frac{c_k(S)e}{E_k}$ (Line 8 and 9). The entire procedure stops when the objective function value is at least one: $\sum_{k=1}^{K} E_k \cdot w_k \ge 1$ (Line 4). Finally, scaling the final flow by $\log_{1+\epsilon} 1/\beta$ yields a feasible solution to the problem (Line 13).

3.3. Analysis

Lemma 1. **AggregationTree** terminates after at most $K \log_{1+\epsilon} \frac{1+\epsilon}{\beta}$ iterations.

Proof: At the beginning of the algorithm, $w_k = \beta/E_k$. The last time the weight of a node is updated, it is less than $1/E_k$, and is increased by at most factor of $1+\epsilon$. Since every iteration the weight of some node is augmented by a factor of at least $1+\epsilon$, the number of such augmentations is at most $K \log_{1+\epsilon} \frac{1+\epsilon}{\beta}$. □

Lemma 2. Scaling the final flow by $\log_{1+\epsilon} \frac{1+\epsilon}{\beta}$ yields a feasible primal solution.

Proof: In the ith iteration of the algorithm, we route certain amount of commodity through a node s_k. Its corresponding energy consumption increases by a fraction $0 \le \gamma^{(i)} \le 1$ of its energy reserve E_k. Its node weight w_k is multiplied by $1+\epsilon\gamma^{(i)}$. Since $(1+\epsilon\gamma^{(i)}) \ge (1+\epsilon)^{\gamma^{(i)}}$ when $0 \le \gamma^{(i)} \le 1$, we have $\prod_i (1+\epsilon\gamma^{(i)}) \ge (1+\epsilon)^{\sum_i \gamma^{(i)}}$. Thus, every time the energy consumption on s_k increases by its energy reserve E_k, its weight w_k increases by a factor of at least $(1+\epsilon)$. Since w_k is initialized as β/E_k, and ends up at most $(1+\epsilon)/E_k$, its total energy consumption cannot exceed $E_k \log_{1+\epsilon} \frac{1+\epsilon}{\beta}$. □

Theorem 1. When $\beta = \frac{1+\epsilon}{((1+\epsilon)K)^{1/\epsilon}}$, **AggregationTree** computes a $(1-2\epsilon)$-approximation to the optimal value of **S** in time $O(\frac{K}{\epsilon^2} \log K \cdot T_{spt})$, T_{spt} being the running time of the shortest path tree algorithm.

Proof: We first prove the approximation property of our algorithm. We make the following denotations. Regarding a set of cost weight assignments w_k $(k = 1, \ldots, K)$, the objective function of $\mathbf{D}(\mathbf{S})$ is $L^{w_k} \triangleq \sum_{k=1}^{K} w_k \cdot E_k$. S^{w_k} is the shortest path tree, which is determined by w_k, and $w(S^{w_k}) \triangleq \sum_{k=1}^{K} c_k(S^{w_k}) \cdot w_k$ is the cost of S^{w_k}.

The objective of $\mathbf{D}(\mathbf{S})$ is to minimize L^{w_k}, subject to the constraint that $w(S^{w_k}) \ge 1$. This constraint can be easily satisfied if we scale the weight of all nodes by $1/w(S^{w_k})$. So $\mathbf{D}(\mathbf{S})$ is equivalent to finding a set of node weights, such that $\frac{L^{w_k}}{w(S^{w_k})}$ is minimized. Thus the optimal value of $\mathbf{D}(\mathbf{S})$ is $OPT \triangleq \min_{w_k} \frac{L^{w_k}}{w(S^{w_k})}$.

In each iteration of the algorithm, the weight of a node s_k is updated. We use $w_k^{(i)}$ to denote the weight of s_k after the ith iteration. $w_k^{(0)} = \beta$ is the initial weight of s_k. Regarding $w_k^{(i)}$, we simplify the following denotations $L^{w_k^{(i)}}$, $S^{w_k^{(i)}}$, and $w(S^{w_k^{(i)}})$, into $L^{(i)}$, $S^{(i)}$ and $w(S^{(i)})$. We also denote $f^{(i)}$ as the total flow that has been routed after the ith iteration. Then based on the node weight update function (Line 9 in Table 5), we have

$$\begin{aligned} L^{(i)} &= \sum_{k=1}^{K} w_k^{(i-1)} \cdot E_k + \epsilon \sum_{k \in S^{(i)}} w_k^{(i-1)} c_k(S^{(i-1)}) e \\ &= L^{(i-1)} + \epsilon(f^{(i)} - f^{(i-1)}) w(S^{(i-1)}) \end{aligned}$$

Since $OPT \le \frac{L^{(i-1)}}{w(S^{(i-1)})}$,

$$\begin{aligned} L^{(i)} &\le L^{(i-1)}(1 + \epsilon(f^{(i)} - f^{(i-1)})/OPT) \\ &\le L^{(i)} e^{\epsilon(f^{(i)} - f^{(i-1)})/OPT} \end{aligned}$$

Thus,

$$L^{(i)} \le L^{(0)} e^{\epsilon f^{(i)}/OPT} \le \beta K \cdot e^{\epsilon f^{(i)}/OPT}$$

The algorithm stops when $L^{(i)}$ reaches 1. Let f^* be the total flow routed, we have

$$1 \le \beta K \cdot e^{\epsilon f^*/OPT}$$

Hence

$$\frac{OPT}{f^*} \le \frac{\epsilon}{\ln(\frac{1}{\beta K})}$$

By Lemma 2, $\frac{f^*}{\log_{1+\epsilon} \frac{1+\epsilon}{\beta}}$ is a feasible solution to $\mathbf{D}(\mathbf{S})$. Then the ratio between the optimal value of **S** and the result returned by our algorithm is

$$\begin{aligned} \frac{OPT}{f^*} \log_{1+\epsilon} \frac{1+\epsilon}{\beta} &\le \frac{\epsilon \log_{1+\epsilon} \frac{1+\epsilon}{\beta}}{\ln\left(\frac{1}{\beta K}\right)} \\ &= \frac{\epsilon \ln \frac{1+\epsilon}{\beta}}{\ln(1+\epsilon)\ln(\frac{1}{\beta K})} \end{aligned} \tag{12}$$

When $\beta = \frac{1+\epsilon}{((1+\epsilon)K)^{1/\epsilon}}$, $\frac{\ln \frac{1+\epsilon}{\beta}}{\ln(\frac{1}{\beta K})} = \frac{1}{1-\epsilon}$. Then we have

$$\begin{aligned} (12) = \frac{\epsilon}{(1-\epsilon)\ln(1+\epsilon)} &\le \frac{\epsilon}{(1-\epsilon)(\epsilon^2 - \epsilon/2)} \le \frac{1}{(1-\epsilon)^2} \\ &\le \frac{1}{1-2\epsilon} \end{aligned}$$

which completes the proof to the approximation property of our algorithm.

Now we prove the second part of the theorem: running time. By Lemma 1, the algorithm terminates after at most $K \log_{1+\epsilon} \frac{1+\epsilon}{\beta}$ rounds. Each round contains a shortest-path tree construction. When $\beta = \frac{1+\epsilon}{((1+\epsilon)K)^{1/\epsilon}}$, it becomes

$$\begin{aligned} K \log_{1+\epsilon}((1+\epsilon)K)^{1/\epsilon} &= \frac{K}{\epsilon}(1 + \log_{1+\epsilon} K) \\ &= \frac{K}{\epsilon}\left(1 + \frac{\log K}{\log(1+\epsilon)}\right) \\ &\le \frac{K}{\epsilon} + \frac{K}{\epsilon^2} \log K \end{aligned}$$

The last inequality holds because $\log(1+\epsilon) \ge \epsilon$ when $\epsilon \le 1$. Thus, the running time of our algorithm is $O(\frac{K}{\epsilon^2} \log K) \cdot T_{spt}$, with $T_{spt} = |\mathcal{L}| + K \log K$ under Dijkstra's algorithm. □

From Theorem 1, it is obvious that our aggregation-tree-based algorithm reduces the running time of Garg-Konemann algorithm by K.

4. Multi-sink data aggregation

Now we proceed to study the multi-sink data aggregation problem. Originally proposed for single-sink data aggregation, our algorithm in Table 5 can be enhanced to accommodate multiple data sinks. Consider that there are M data sinks in the network. Each commodity consists of a source s_i, and M data sinks $d^1, d^2, \ldots, d^M$. s_i can split its commodity and send to any of the data sink in parallel. In other words, each data sink may only receive a subset of s_i's data. It is up to the data sinks to communicate with each other and recover the entire content from each of their own pieces.

If we denote $\mathcal{P}_i = \{P_j^i\}$ as the set containing all paths from s_i to $d^1, d^2, \ldots, d^M$, then this problem has the same formulation as **P**, therefore can be addressed using the algorithm **MaxConcurrentFlow** in Section 2.2, with a slight modification in Line 8. In order to get the shortest path in $\mathcal{P}_i$, one has to first compute the shortest paths from s_i to $d^1, d^2, \ldots, d^M$ separately, then picks the shortest one among them. Alternatively, we can compute the shortest path tree that is rooted at s_i, and spanning $d^1, d^2, \ldots, d^M$ as its leaves. In this way, the running time of the algorithm stays unchanged.

Now we see how our aggregation tree formulation can provide an efficient algorithm for multi-sink problem. We enhance the concept of *aggregation tree* as follows. Consider the unification of K paths, each from s_i $(i = 1, \ldots, K)$ to one of the sinks $d^q (q = 1, \ldots, M)$. We temporarily call such a graph a *multi-source aggregation tree*, denoted as R_j. The set of such graphs is denoted as $\mathcal{R} = \{R_j\}$. Later in this section, we will show that R_j is actually a forest consisting of M trees rooted at $d^1, d^2, \ldots, d^M$ separately. Moreover, for R_j, the flow rates on each of its paths $P_j^i (i = 1, \ldots, K)$ are in proportion to their demands based on Equation (9). The flow rate of R_j is the aggregated rate of all K paths, i.e., $f(R_j) = \sum_{i=1}^{K} f(P_j^i)$. The problem is formulated as

$$\begin{aligned} \mathbf{R}: \ \text{maximize} & \sum_{j=1}^{|\mathcal{R}|} f(R_j) \\ \text{subject to} & \sum_{j=1}^{|\mathcal{R}|} c_k(R_j) \cdot f(R_j) \le E_k, \quad k = 1, \ldots, K \\ & f(R_j) \ge 0, \forall j \end{aligned} \tag{13}$$

Here $c_k(R_j)$ is defined in the same fashion as $c_k(S_j)$ in Section 3.

$$c_k(R_j) \triangleq \frac{\sum_{i=1}^{K} D_i \cdot c_k(P_j^i)}{\sum_{i=1}^{K} D_i}$$

Table 6
Notations in Section 4.

Notation	Description
$\mathcal{R} = \{R_j\}$	Set of aggregation forests from all sources s_i $(i = 1, \ldots, K)$ to destinations $d_1, \ldots, d^M$
$f(R_j)$	Amount of data flow sent via R_j
$c_k(R_j)$	Energy cost of node s_k if one unit of data is sent via R_j

Table 7
Algorithm for the multi-sink data aggregation problem.

AggregationForest

1 $k = 1, \ldots, K, w_k \leftarrow \beta / E_k$
2 $\forall i, j, k = 1, \ldots, K, c_k(P_j^i) \leftarrow w_k \cdot c_k(P_j^i)$
3 $f(R_j) \leftarrow 0, R_j \in \mathcal{R}$
4 **while** $\sum_{k=1}^{K} E_k \cdot w_k < 1$
5 $R \leftarrow$ unification of K shortest paths, each from s_i $(i = 1, \ldots, K)$ to any of the destinations $d^1, d^2, \ldots, d^M$, with the cost of link (n, n'), as $c^t(n, n') + c^r$
6 $e \leftarrow \min_{s_k \in R} \frac{E_k}{c_k(R)}$
7 $f(R) \leftarrow f(R) + e$
8 **for** $\forall s_k \in R$ **do**
9 $w_k \leftarrow w_k(1 + \epsilon \frac{c_k(R)e}{E_k})$
10 $\forall s_k \in R, c_k(R) \leftarrow c_k(R)(1 + \epsilon \frac{c_k(R)e}{E_k})$
11 **end for**
12 **end while**
13 Scale $f(R_j)$ by $\log_{1+\epsilon} 1/\beta, \forall j$

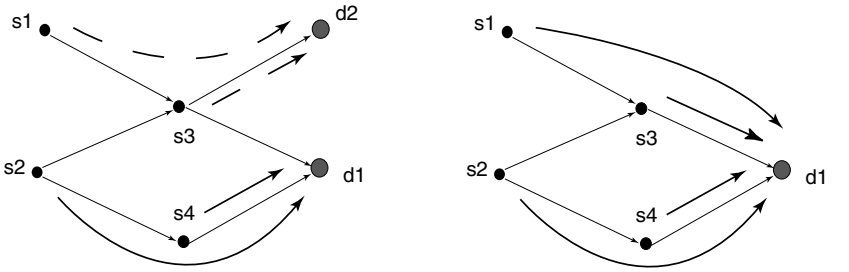
Table 6 lists the new notations introduced in this section. Similar to **S**, we can show that **P** and **R** are equivalent regarding the problem of multi-sink data aggregation. The dual problem of **R** is given as follows.

$$\begin{aligned} \mathbf{D(R)}: \ \text{minimize} & \sum_{k=1}^{K} E_k \cdot w_k \\ \text{subject to} & \sum_{k=1}^{K} c_k(R_j) \cdot w_k \ge 1, \quad j = 1, \ldots, |\mathcal{R}| \\ & w_k \ge 0, k = 1, \ldots, K \end{aligned} \tag{14}$$

D(R) has the same physical meaning as **D(S)**. The polynomial-time algorithm to D(R) is listed in Table 7.

The similarity between algorithms **AggregationTree** and **AggregationForest** is obvious. In fact, the only major difference between these two algorithms lies in Line 5, i.e., the way the aggregation tree is computed, as shown in figure 3. Now we show that in **AggregationForest**, the unification of K shortest paths, each from a source s_i $(i = 1, \ldots, K)$, actually forms a forest consisting of M trees rooted at $d^1, d^2, \ldots, d^M$ separately.

To prove this, we only need to show the following fact. Without loss of generality, for source s_1 whose shortest path



(a) In multi-sink case, in each iteration a forest is formed

(b) In single-sink case, in each iteration a tree is formed

Figure 3. Aggregation forest vs. aggregation tree.

leads to the destination d^1, if another source s_2 appears on this path, then the shortest path of s_2 also leads to d^1. In other words, the shortest paths of s_1 and s_2 share the same postfix. Otherwise, if the shortest-path destination of s_2 is another one, say d^2, then the length of the path from s_1 to s_2, then to d^2 would be shorter than the current path from s_1 to d^1, contradicting the fact this path is already the shortest one.

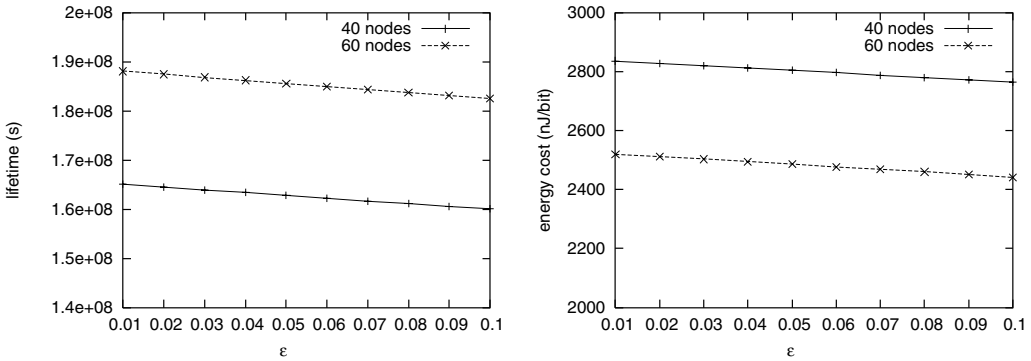
We can solve the **AggregationForest** problem by slightly modifying Dijkstra's shortest path tree algorithm. In the original algorithm, an index value is attached to each node, denoted as $key(s)$, initialized as ∞, except for the root node, whose index value is 0. In each iteration, the node with the smallest index value is chosen as the new tree node, say s. Then for each next-hop neighbor of s that has not been chosen yet, say s', its index value is updated as $key\ (s') \leftarrow \min\ (key\ (s'), key\ (s) + c^t(s, s') + c^r)$. The algorithm stops when all nodes have been chosen. To accommodate this algorithm to our problem, we only need to initialize the index value of all data sinks as 0, namely, $key\ (d^1) = \cdots = key\ (d^M) = 0$, and arbitrarily break the tie when choosing the node with the smallest index value during the algorithm. The modified algorithm has the same performance bound as the original one. Therefore, all analytical results for the algorithm **AggregationTree** in Section 3.3 hold for **AggregationForest**.

5. Simulation studies

5.1. Experimental setup

We evaluate the performance of our algorithm via simulation in this section. Our simulation setting is as follows. We randomly create n (ranged from 30 to 100) nodes on a $100 \times 100\ \mathrm{m}^2$ square. The data sinks are also randomly located within this square. The data generating rate of each sensor is 0.5 Kbps. The maximum transmission range of each sensor is 25 m. In our experiment, we set $\alpha = 50\ \mathrm{nJ/b}$, $\beta = 0.0013\ \mathrm{pJ/b/m}^4$ and $m = 4$ for the power consumption model. The energy reserve on each sensor is 50 kJ. We run each experiment over 20 different random topologies. For example, when evaluating the lifetime of the network with 40 sensors, we create 20 different 40-node topologies and run algorithms on each of them, then show the average result.

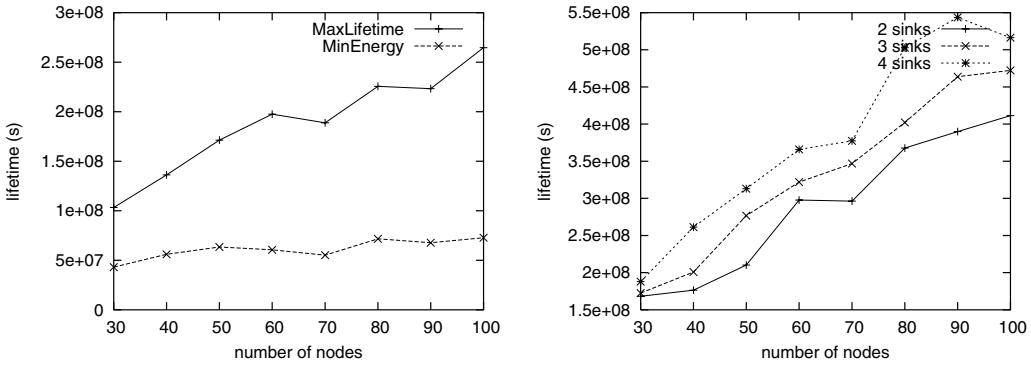
We compare the performance of our algorithm (**MaxLife**) with the minimum energy routing (**MinEnergy**) algorithm, whose target is to minimize the energy consumption for each data unit routed through the network. For each data source, the algorithm finds its shortest path to the data sink in terms of energy cost. The route for each data source is fixed throughout the entire network lifetime.



(a) Network Lifetime

(b) Energy Cost

Figure 4. Impact of ϵ (Single Sink).



(a) Single Sink (b) Multiple Sinks (MaxLife)

Figure 5. Network lifetime.

5.2. Impact of ϵ

First we show the impact of ϵ, the parameter of the approximation algorithm, on the network performance. From figure 4, we observe that large value of ϵ will slightly decrease the network lifetime. This validates the approximation property given in Theorem 1. Moreover, we also observe that the energy consumption per bit also slightly decreases with the increase of ϵ. This interesting results show that to achieve longer network lifetime, data may need to be routed in an energy-inefficient way, thus consume more energy.

5.3. Network lifetime

Figure 5(a) shows the lifetime of the same network when the data is routed by different algorithms under the single-sink setting. We observe that as the network size grows, our algorithm is able to at least double the lifetime returned by the **MinEnergy** algorithm. From figure 6, we also notice that when the network size further grows, the network lifetime stops increasing. Our explanation for such a phenomenon is as follows. When the topology turns from sparse to dense, the average distance between neighboring nodes decreases, which means a node needs less power to send a data unit to its neighbor. This is the primary reason for the quick network lifetime growth when $n \leq 100$. On the other hand, as we deploy more nodes, the density of the network topology increases, so does the density of the traffic, since each node is a data source. This results in the slight lifetime decrease when $n \geq 200$. Figure 5(b) further shows that when more data sinks exist in the network, the network lifetime increases. This result is obvious, as more data can be routed to the sink via a shorter path when more data sinks are present. Such gain in network lifetime increases with the number of nodes when $n \leq 100$.

Figure 7 shows the lifetime ratio of **MaxLife** algorithm to **MinEnergy** algorithm. The result shows that **MaxLife** algorithm outperforms **MinEnergy** algorithm at different network sizes, and different numbers of data sinks. Moreover, such a performance gain tends to decrease when the number of data sinks grows.

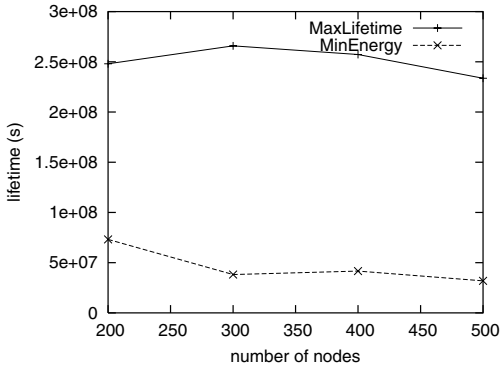


Figure 6. Network lifetime in dense network.

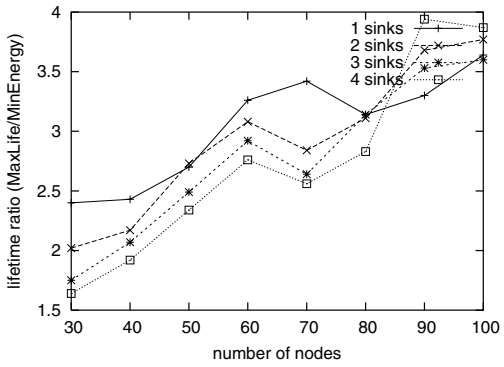
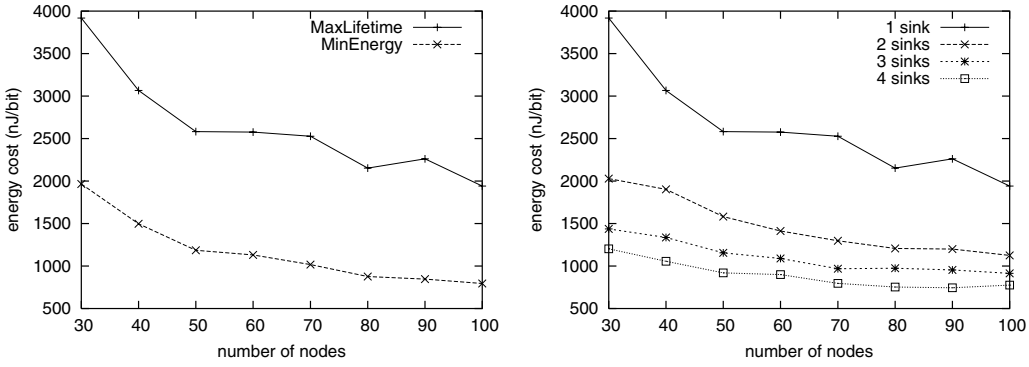


Figure 7. Network lifetime of MaxLife to MinEnergy.

5.4. Energy cost

On the other hand, as shown in figure 8, our algorithm consumes more energy than **MinEnergy** for an average bit of data routed through the network. The reason is that, in order to maximally utilize the energy reserve of all sensor within the network, sometimes the data from a source has to go through some route whose energy consumption rate is not as efficient as the one returned by **MinEnergy** algorithm. In figure 9, we see that the average energy consumption rate of our algorithm is more than two times the energy consumption rate of



(a) Single Sink (b) Multiple Sinks (MaxLife)

Figure 8. Energy cost.

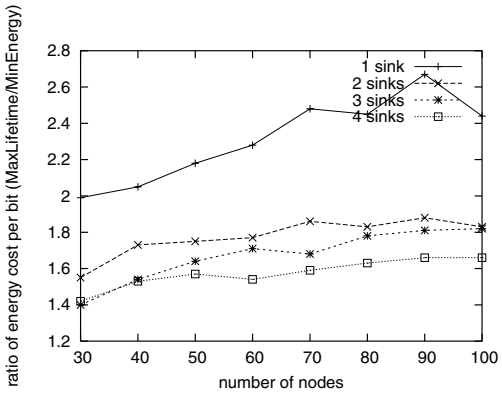


Figure 9. Energy cost ratio of MaxLife to MinEnergy.

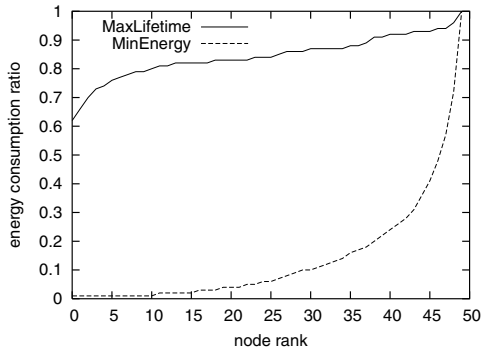


Figure 10. Energy consumption distribution (50 Nodes, Single Sink).

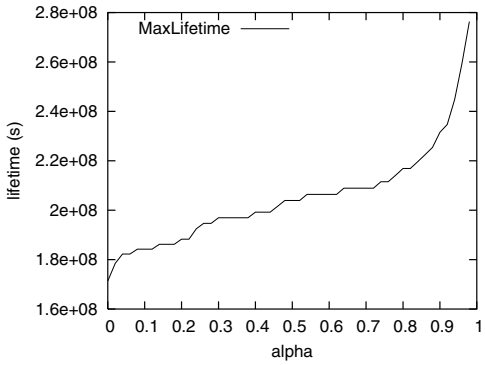


Figure 11. α-lifetime (50 Nodes, Single Sink).

MinEnergy. Such a ratio quickly drops to around 1.5 when the number of sinks is more than one.

5.5. Distribution of energy consumption

The distinction of two algorithms' energy consumption pattern is further exhibited in figure 10, which plots the distribution of the energy consumption ratio of each node in the network throughout the entire lifetime. For **MinEnergy** algorithm, only a few "hot spot" nodes completely utilize their energy reserves. Obviously, they are the ones located closed to the data sink. In other words, these nodes are at the top of the fixed data aggregation tree returned by the algorithm, where each of them has to relay heavy amount of traffic for nodes from its subtree. Meanwhile, those nodes located at the lower levels of the tree only contribute little of their energy reserves before the network lifetime expires.

On the other hand, in the result of our algorithm, most nodes get to contribute about 80% of their energy reserve, since our algorithm always tends to route traffic via the least-loaded node in terms of the remained energy reserve. Thus, our algorithm is able to sustain a much longer system lifetime at the price of more energy consumption per bit than the **MinEnergy** algorithm.

As sensor networks are usually used for data collection, the network may function with a lower precision of the collected data, when a portion of the sensors run out of energy. α-lifetime, which is defined as the time when an α portion of the sensors drain their energy, can better reflect the characteristic of such sensor network functionalities. Figure 11 plots the averaged α-lifetime of 50-node sensor networks under our algorithm. The results are consistent with the observations on energy consumption distribution, where α-lifetime increases slowly with α when it is less than 0.8.

6. Related works

In this section, we review the existing literatures and compare our work with the existing works.

In the existing works, the problem of designing energy-efficient routing algorithms has been extensively studied in both general multihop wireless networks [5,6,18,23–25], and the particular backdrop of sensor networks [9–12]. These energy-efficient routing algorithms can achieve the goal of either minimizing energy consumption [7,14–19], or maximizing the network lifetime [5,6,8,20,24], or maximizing the network capacity [18].

To achieve the goal of minimum energy routing, the typical approach [13,22,27–29] is to use a shortest path algorithm in which the edge cost is related to the power required to transmit a packet between two nodes of this edge. The problem with this approach is that it causes unbalanced power consumption. And nodes on the minimum-energy path are quickly drained of power. Though some routing algorithms [24,26] associate a cost with the node of low energy reserve, they remain a heuristic solution.

On the other hand, our work addresses the problem of maximizing network lifetime which well addresses the power consumption balance problem. The works of [5,6,8,20,23] formulate the problem of maximizing network lifetime using linear programming. Based on this formulation, Chang and Tassiulas [5] present a heuristic algorithm to solve the linear program approximately. In [6], they further give a centralized algorithm based on the Garg-Koenemann [12] algorithm for multicommodity flow to determine the maximum lifetime. In the work of [8], Kalpakis et al. present a heuristic algorithm, whose running time has a bad scalability to the network size. Compared with these works, our algorithm scales well in terms of network size. By making use of the traffic characteristic of data aggregation, our algorithm improves the algorithm running time by K, K being the number of commodities. We compare the performance of our algorithm with the existing algorithms [5,6,8] in Table 8. In the table, two metrics are compared, namely ρ, the ratio to optimum, and the running time. Here ρ is defined as the ratio between the worst case network lifetime of a particular algorithm and the optimal network lifetime. Among these algorithms, the flow augmentation algorithm $FA(x_1, x_2, x_3)$ in [5] does not have an analytical result with respect to its input parameters. The data listed in the table are among its best reported results with augmentation step size as 0.001, and $x_1 = 1, x_2 = 50, x_3 = 50$. ρ of this algorithm can be much smaller, if the step size is larger, or x_2 and x_3 take different values. The comparison results show that our algorithm outperforms the existing algorithms in time complexity. Sankar and Liu [23] formulate the problem as a maximum concurrent flow problem and adopt a distributed flow algorithm [2,3]. Yet, this algorithm can only verify whether a traffic demand can be satisfied. To calculate the exact network lifetime and conduct maximum lifetime routing, a bisection search is needed. As the algorithm has a long running time, it can lead to very slow convergence in routing, when combined with bisection search.

Table 8
Performance Comparison. In the table, T_{spt} is the running time of the shortest path tree algorithm. T_{sp} is the running time of the shortest path algorithm. These two runing times have the same complexity under Dijkstra's algorithm.

Algorithrn	Ratio to optimum ρ	Running time
AggregationTree	$(1-2\epsilon)$	$O(\frac{K}{\epsilon^2}\log K \cdot T_{spt})$
$FA(1, 50, 50)$ [5]	0.99	N/A
Garg-Koenemann [6]	$(1-3\epsilon)$	$O(\frac{K^2 \log K}{\epsilon^2} \cdot T_{sp})$
MLDA[8]	$1-\epsilon$	$O(K^{15}\log K)$

Hou et al. [15] present an algorithm for max-min node lifetime problem for sensor networks. The goal of this algorithm is to achieve lexicographic max-min node lifetime distribution instead of maximizing the lifetime of the first node which is drained of energy. They also present in [16] a polynomial-time algorithm based on parametric analysis to achieve max-min rate allocation and prove an interesting result–the duality relationship between rate allocation problem and node lifetime problem. The work of [25] also presents a rate allocation algorithm based on traffic splits in ad hoc networks. Our algorithm addresses a different problem from these works and may fit in different network operation environments. In particular, in our problem, the traffic demands from all sensors are fixed and known a priori. This problem well models the application scenarios, such as temperature, pressure, noise level monitoring, where fixed amount of information is generated at a fixed interval. Our goal is to maximize the network lifetime while satisfying the rate demands, instead of allocating rates to different sensors such that certain fairness criteria are satisfied. Moreover, the work of [11] also studies the problem of improving the network lifetime with multiple base stations. Yet, their solution remains a heuristic one.

7. Conclusion

This paper studies the problem of maximizing lifetime routing for data aggregation in sensor networks. Inspired by Garg-Konemann algorithm for multicommodity flow, this paper presents a novel tree-based approximation algorithm, which addresses the unique traffic characteristic of data aggregation and achieves faster running time. Using this algorithm, this paper further studies the impact of multiple data sinks on the lifetime of sensor networks, and presents interesting observations.

References

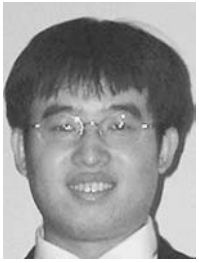
[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cyirci, Wireless sensor networks: A survey, *Computer Networks* 38(4), (2002) 393–422.

[2] B. Awerbuch and T. Leighton, A simple local-control approximation algorithm for multicommodity flow, in: *Proc. of 3rd IEEE Symposium on Found. of Comp. Science* (1993) pp. 459–468.

[3] B. Awerbuch and T. Leighton, Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in

dynamic networks, in: *Proc. of the 26th Annual ACM Symposium on Theory of Computing* (1994) pp. 487–496.
[4] M. Bhardwaj and A.P. Chandrakasan, Bounding the lifetime of sensor networks via optimal role assignments, in: *Proc. of INFOCOM* (2002) pp. 1587–1596.
[5] J. Chang and L. Tassiulas, Energy conserving routing in wireless ad-hoc networks, in: *Proc. of INFOCOM* (2000) pp. 22–31.
[6] J. Chang and L. Tassiulas, Fast approximate algorithms for maximum lifetime routing in wireless ad-hoc networks, in: *Proc. of NETWORKING* (2000) pp. 702–713.
[7] X. Cheng, B. Narahari, R. Simha, M.X. Cheng and D. Liu, Strong minimum energy topology in wireless sensor networks: NP-completeness and heuristics, IEEE Tran. on Mobile Computing 2(3) (2003).
[8] K. Dasgupta, K. Kalpakis and P. Namjoshi, Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks, Computer Networks 42 (2003) 697–716.
[9] K. Dasgupta, M. Kukreja and K. Kalpakis, Topology-aware placement and role assignment for energy-efficient information gathering in sensor networks, in: *Proc. of the 8th IEEE Symposium on Computers and Communications* (2003).
[10] L.K. Fleischer, Approximating fractional multicommodity flow independent of the number of commodities, *SIAM Journal of Discrete Mathematics* 13 (2000) 505, 520.
[11] S.R. Gandham, M. Dawande, R. Prakash and S. Venkatesan, Energy-efficient schemes for wireless sensor networks with multiple mobile base stations, in: *Proc. of IEEE Globecom* (2003).
[12] N. Garg and J. Konemann, Faster and simpler algorithms for multicommodity flow and other fractional packing problems, in: Proc. of IEEE Symposium on Found. of Comp. Science (1998) pp. 300–309.
[13] J. Gomez, A. Campbell, M. Naghshineh and C. Bisdikian, Conserving transmission power in wireless ad hoc networks, in: *Proc. of 9th International Conference on Network Protocols (ICNP)* (2001).
[14] M. Grotschel, L. Lovasz and A. Schrijver, Geometric Algorithms and Combinatorial Optimizations, Springer-Verlag (1993).
[15] Y.T. Hou, Y. Shi and H.D. Sherali, On lexicographic max-min node lifetime problem for energy-constrained wireless sensor networks, in: *Technical Report, The Bradley Dept. of ECE, Virginia Tech* (2003).
[16] Y.T. Hou, Y. Shi, and H.D. Sherali, On rate allocation problem for wireless sensor networks, in: *Technical Report, The Bradley Dept. of ECE, Virginia Tech* (2003).
[17] C. Intanagonwiwat, R. Govindan and D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in: *Proc. of MOBICOM* (2000).
[18] K. Kar, M. Kodialam, T.V. Lakshman and L. Tassiulas, Routing for network capacity maximization in energy-constrained ad-hoc networks, in: *Proc. of INFOCOM* (2003).
[19] B. Krishanamachari, D. Estrin and S. Wicker, The impact of data aggregation in wireless sensor networks, in: *Proc. of International Workshop of Distributed Event Based Systems (DEBS)* (2002).
[20] Q. Li, J. Aslam and D. Rus, Online power-aware routing in wireless ad-hoc networks, in: *Proc. of MOBICOM* (2001) pp. 97–107.
[21] S. Lindsey, C.S. Raghavendra and K. Sivalingam, Data gathering algorithms in sensor networks using energy metrics, *IEEE Trans. on Parallel and Distributed Systems* (2002).
[22] V. Rodoplu and T. Meng, Minimum energy mobile wireless networks, IEEE Journal of Selected Areas in Communications 17(8) (1999) 1333–1344.
[23] A. Sankar and Z. Liu, Maximum lifetime routing in wireless ad-hoc networks, in: *Proc. of INFOCOM* (2004).
[24] S. Singh, M. Woo and C. Raghavendra, Power-aware routing in mobile ad-hoc networks, in: *Proc. of MOBICOM* (1998) pp. 181–190.
[25] V. Srinivasan, C.F. Chiasserini, P. Nuggehalli and R.R. Rao, Optimal rate allocation and traffic splits for energy efficient routing in ad hoc networks, in: *Proc. of INFOCOM*, 2002.
[26] I. Stojmenovic and X. Lin, Power-aware localized routing in wireless networks, *IEEE Tran. on Parallel and Distributed Systems* 12(11) (2001) 1122–1133.
[27] R. Wattenhofer, L. Li, P. Bahl and Y. Wang, Distributed topology control for power efficient operation in multihop wireless ad hoc networks, in: *Proc. of INFOCOM* (2001).
[28] Y. Xue and B. Li, Location-aided power-aware routing for wireless ad-hoc networks, in: *Proc. of Globecom* (2001).
[29] G. Zussman and A. Segall, Energy efficient routing in ad hoc disaster recovery networks, in: *Proc. of INFOCOM* (2003).

Yuan Xue received her B.S. in Computer Science from Harbin Institute of Technology, China in 1994 and her M.S. and Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign in 2002, and 2005. Currently she is an assistant professor at the Department of Electrical Engineering and Computer Science of Vanderbilt University. Her research interests include wireless and sensor networks, mobile systems, and network security.
E-mail: yuan.xue@vanderbilt.edu



Yi Cui received his B.S. and M.S. degrees in 1997 and 1999, from Department of Computer Science, Tsinghua University, China, and his Ph.D. degree in 2005 from the Department of Computer Science, University of Illinois at Urbana-Champaign. Since then, he has been with the Department of Electrical Engineering and Computer Science at Vanderbilt University, where he is currently an assistant professor. His research interests include overlay network, peer-to-peer system, multimedia system, and wireless sensor network.
E-mail: yi.cui@vanderbilt.edu

Klara Nahrstedt (M ' 94) received her A.B., M.Sc degrees in mathematics from the Humboldt University, Berlin, Germany, and Ph.D in computer science from the University of Pennsylvania. She is an associate professor at the University of Illinois at Urbana-Champaign, Computer Science Department where she does research on Quality of Service(QoS)-aware systems with emphasis on end-to-end resource management, routing and middleware issues for distributed multimedia systems. She is the coauthor of the widely used multimedia book 'Multimedia:Computing, Communications and Applications' published by Prentice Hall, and the recipient of the Early NSF Career Award, the Junior Xerox Award and the IEEE Communication Society Leonard Abraham Award for Research Achievements, and the Ralph and Catherine Fisher Professorship Chair. Since June 2001 she serves as the editor-in-chief of the ACM/Springer Multimedia System Journal.
E-mail: klara@cs.uiuc.edu