Data Gathering in Wireless Sensor Networks with Mobile Collectors

Ming Ma and Yuanyuan Yang

Department of Electrical and Computer Engineering, State University of New York, Stony Brook, NY 11794, USA

***Abstract*—In this paper, we propose a new data gathering mechanism for large scale wireless sensor networks by introducing mobility into the network. A mobile data collector, or for convenience called *M-collector* in this paper, could be a mobile robot or a vehicle equipped with a powerful transceiver and battery, works like a mobile base station and gathers data while moving through the field. An M-collector starts the data gathering tour periodically from the static data sink, polls each sensor while traversing the transmission range of the sensor, then collects data directly from the sensor without relay (i.e., in a single hop), finally returns and uploads data to the data sink. Since data packets are gathered directly without relay and collision, the lifetime of sensors is expected to be prolonged, and sensors can be made very simple and inexpensive. We focus on the problem of minimizing the length of each data gathering tour and refer to this problem as the single-hop data gathering problem, or SHDGP for short. We first formalize the SHDGP problem into a mixed integer program and prove its NP-hardness. We then present a heuristic tour-planning algorithm for a single M-collector. For the applications with a strict distance/time constraint for each data gathering tour, we also utilize multiple M-collectors to traverse through several shorter sub-tours concurrently to satisfy the distance/time constraint. Our single-hop data gathering scheme will improve the scalability and solve intrinsic problems of large scale homogeneous networks, and can be used in both connected networks and disconnected networks. The simulation results demonstrate that the new data gathering algorithm can greatly reduce the moving distance of the collectors compared to the *covering line* approximation algorithm, and is close to the optimal algorithm in small networks. In addition, the proposed data gathering scheme can prolong the network lifetime significantly compared to a network which has only a static data collector, or a network in which the mobile collector can only move along straight lines.**

Keywords: Mobile data collector, M-collector, Wireless sensor networks, Data gathering, Movement planning.

I. Introduction and Previous Work

In recent years, wireless sensor networks have emerged as a new information-gathering paradigm in a wide-range of applications, such as medical treatment, outer-space exploration, battlefield surveillance, emergency response, etc. [1]-[15]. Sensor nodes are usually thrown into a large scale sensing field without a pre-configured infrastructure. Before monitoring the environment, sensor nodes must be able to discover nearby nodes, organize themselves into a network. Most of the energy of a sensor is consumed on two major tasks: sensing the field and uploading data to the data sink. Energy consumption on sensing is relatively stable because it only depends on the sampling rate, and does not depend on the network topology or the location of sensors. On the other hand, the data gathering scheme is the most important factor that determines the network lifetime. Although the applications of sensor networks may be quite diverse, most of them share a common feature: Their data packets may need to be aggregated at some data sink. In a homogeneous network where sensors are organized into a flat topology, sensors close to the data collector consume much more energy than sensors at the margin of the network, since they need to relay a lot of packets from sensors far away from the data collector. As a result, after these sensors fail, other sensors cannot reach the collector and the network becomes disconnected, even most of the nodes can still survive for a long period. Therefore, for a large-scale, data-centric sensor network, it is inefficient to use a single, static data sink to gather data from all sensors. In some applications, sensors are deployed to monitor separate areas. In each area, sensors are densely deployed and connected, while sensors that belong to different areas may be disconnected. Unlike fully connected networks, some sensors cannot forward data to the data sink via wireless links. A mobile data collector is perfectly suited to such applications. A mobile data collector serves as a mobile "data transporter" which moves through every community and links all separated subnetworks together. The moving path of the mobile data collector acts as virtual links between separated subnetworks.

Due to the limitation of the flat topology, some previous work [16], [17], [18], [19], [20] introduced a hierarchy to the network. In such a network, sensor nodes are organized into clusters and form the lower layer of the network. At the higher layer, cluster heads collect sensing data from sensors and forward data to the outside data sink. Such two-layered hybrid networks are more scalable and energy-efficient than homogeneous sensor networks. A cluster head acts not only as a data aggregation point for collecting sensing data from sensors, but also as a controller/scheduler to make various routing and scheduling decisions. In a homogeneous network, all nodes have identical capability and energy at the beginning. Some of the nodes are selected and serve as cluster heads [16], [17], [18]. However, cluster heads will inevitably consume more energy than other sensor nodes. In order to avoid the problem of cluster heads failing faster than other nodes, sensor nodes can become cluster heads rotationally [16]. In this type of the network, since every sensor node may possibly become a cluster head, each of them has to be "powerful" enough to handle incoming and outgoing traffic and cache sensing data, which will increase the overall cost of the entire sensor network. Furthermore, selecting cluster heads dynamically also results in high overhead due to the frequent information exchange among sensor nodes. Some efforts have been made to improve the intrinsic disadvantage of homogeneous networks by introducing a small number resource-rich nodes. Unlike homogeneous networks, a heterogeneous sensor network contains a small number of resource-rich nodes together with a large number of resource-limited basic sensor nodes. Basic sensor nodes have limited communication capability and mainly focus on sensing the environment, while resource-rich nodes are equipped with more powerful transceivers and batteries. In [19], [20], resource-rich nodes act as cluster heads, where the network can be organized into a two-layered hierarchical network. However, it is difficult to deploy powerful cluster heads nodes to appropriate positions without learning the network topology.

Recently, mobility in sensor networks has received much attention, see, for example, [7], [8], [9], [10], [11], [12], [13], [14]. In [7] and [8], radio-tagged zebras and whales are used as mobile nodes to collect sensing data in a wild environment. These animal-based nodes wander randomly in the sensing field, and

The research work was supported in part by the U.S. National Science Foundation under grant number ECS-0427345 and U.S. Army Research Office under grant number W911NF-04-1-0439.

exchange sensing data only when they move close to each other. Thus, sensor nodes in such a network are not necessarily connected all the time. Moreover, the mobility of randomly moving animals is hard to predict and control, thus the maximum packet delay cannot be guaranteed. For sensor networks deployed in an urbane area, public transportation vehicles such as buses and trains can act as mobile base stations [10], [12]. In this case, the moving path and timing are predictable. However, data exchanging still depends on the existing routes and schedules of the public transportation, and is thus very restrictive. In [13], a number of mobile collectors, called data mules, traverse the sensing field along parallel straight lines and gather data from sensors. This scheme works well in a large scale, uniformly distributed or randomly distributed sensor network. However, in practice, data mules may not always be able to move along straight lines, for example, obstacles or boundaries may block the moving paths of data mules. Moreover, the performance and the cost of the data mule scheme depends on the number of data mules and the distribution of sensors. When only a small number of data mules are available and not all sensors are connected, data mules may not cover all the sensors in the network if they only move along straight lines. [9], [14] also considered mobile collectors in sensor networks. [9] mainly discussed hardware/software implementation of underwater mobile collectors, while [14] proposed an algorithm to schedule the mobile collector, such that there is no data loss due to buffer overflow.

In this paper, we consider applications, such as environment monitoring for human-unreachable environment, where sensing data is generally collected at a low rate and is not so delay-sensitive that it can be accumulated into fixed-length data packets and uploaded once a while. We introduce mobility to the resource-rich nodes, such that the energy efficiency and the scalability of the data gathering scheme can be greatly improved. To elaborate, we consider using one or more mobile data collectors to gather the data from sensors. A mobile data collector could be a mobile robot or a vehicle equipped with a powerful transceiver, battery and large memory. The mobile data collector starts a tour from the data sink, traverses the network, collects sensing data from nearby nodes while moving, and then returns and uploads data to the data sink. Since the data collector is mobile, it can move close to the sensor nodes, such that if the moving path is well-planned the network lifetime can be greatly prolonged. In the following, for convenience, we use **M-collector** to denote the mobile data collector. We propose a new data gathering mechanism with M-collectors for large scale sensor networks, in which the mobile data collectors will collect data from all sensors through one-hop communications.

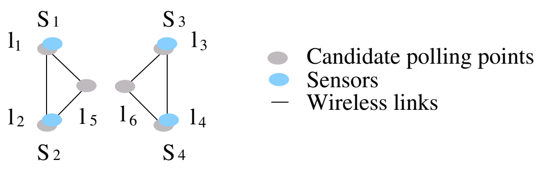
The rest of the paper is organized as follows. Section II gives some assumptions and terms that will be used in the remaining of the paper. In Section III, we formalize the single hop data gathering problem into a mixed integer program and prove its NP-hardness. Section IV presents the new data gathering algorithm for a single M-collector. Section V extends the data gathering algorithm to work in a network with multiple M-collectors. Section VI gives simulation results and some discussions. Finally, Section VII concludes the paper.

II. Preliminaries

In this paper, we consider the data gathering problem in which the M-collector can visit the transmission range of every sensor, such that sensing data can be collected by a single hop communication without relay. Before we formally define the data gathering problem, we first define some terms that will be used in the rest of the paper.

While an M-collector is moving, it can poll nearby sensors one by one to gather data. Upon receiving the polling message, a sensor simply uploads the data to the M-collector directly without relay. We define the positions where the M-collector polls sensors as *polling points*. When an M-collector moves to a polling point, it polls nearby sensors with the same transmission power as sensors, such that sensors that receive the polling messages can upload packets to the M-collector in one hop. After gathering data from sensors around the polling point, the M-collector moves straight to the next polling point in the tour. Thus, each data gathering tour of an M-collector consists of a number of polling points and the straight line segments connecting them. For example, let $P = \{p_1, p_2, \ldots, p_t\}$ denote a set of polling points and DS be the data sink. Then, the moving tour of the M-collector can be represented by $DS \rightarrow p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_t \rightarrow DS$. Thus, the problem of finding the optimal tour can be considered as the problem of determining the locations of polling points and the order to visit them. Before an M-collector starts a data gathering tour, it needs to determine the positions of all polling points, and which sensors it can poll at each polling point. We define *neighbor set* of a point in the plane as the set of sensors that can upload data to the M-collector directly without relay, if the M-collector polls sensors at this point. Since the M-collector can only collect data at polling points, each sensor must be in the neighbor set of at least one polling point to upload data without relay. In other words, the union of neighbor sets of all polling points must cover all sensors.

In most of existing work, the transmission range of an omni-directional antenna was simply assumed to be a disk-shaped area around the transceiver. Based on this assumption, given a point in the plane, the neighbor set of this point consists of all sensors within the disk-shaped area around this point. However, due to the uncertainties of the wireless environment, such as signal fading, reflection from walls and obstacles and interference, it is hard to estimate the boundary of the transmission range without real measurement [21], [22]. Therefore, in practice, it is almost impossible to obtain the neighbor set of an unknown point, unless the M-collector has moved to this point and tested wireless links between it and its one-hop neighbors, or a sensor has been placed at this point and acquired all its one-hop neighbors during the neighbor discovering phase. Thus, it is only possible to test a finite number of points and their corresponding neighbor sets in the plane, and we must select polling points from this finite set of points, which we refer to as the *candidate polling point set*. If the connection pattern of sensors can be obtained, or in other words, we know the one hop neighbors of each sensor, the position of each sensor can be a candidate polling point, since the neighbor set of this point is already known. However, the connection pattern may not always be available before sending out M-collectors, unless the network is completely connected so that the connection pattern can be reported to the data sink via



Sensors : S_1, S_2, S_3 and S_4

Neighbor set of l_1, l_2 and l_5 : { S_1, S_2 }

Neighbor set of l_3, l_4 and l_6 : { S_3, S_4 }

Candidate polling point set : { l_1, l_2, l_3, l_4, l_5, l_6 }

Fig. 1. Examples of *polling points*, *neighbor sets*, and *candidate polling point set*.

wireless transmissions. In order to obtain the candidate polling points without the information on the connection pattern, after sensors are deployed, one or more M-collectors need to explore the entire sensing field. While exploring, each M-collector can broadcast "Hello" messages periodically with the same transmission power as sensors. Sensor that can decode "Hello" message correctly replies an "ACK" message to notify the M-collector where it is. Upon receiving the "ACK" message from the sensor, the M-collector marks its current location as a candidate polling point and adds the ID of the sensor into the neighbor set of this candidate polling point. Thus, all wireless links between sensors and the M-collector at the candidate polling points are bi-directionally tested. In addition, each sensor can also discover its one-hop neighbors by broadcasting the "Hello" messages during the neighbor discovering phase. After the sensor reports the IDs of its one-hop neighbors to the M-collector by including the information into the "ACK" message, the position of the sensor can also become a candidate polling point. In Fig. 1, we illustrate the definition of *polling points*, *neighbor set*, and *candidate polling point set* by an example, where there are four sensors s_1, s_2, s_3 and s_4 deployed at the position l_1, l_2, l_3 and l_4, respectively. During the exploration phase, the M-collector discovers the neighbor sets of l_5 and l_6 by broadcasting "Hello" messages at these points. Thus, l_5 and l_6 can be added into the candidate polling point set. Since sensors s_1, s_2, s_3 and s_4 also report their one-hop neighbors to the M-collector by sending "ACK" to the M-collector, l_1, l_2, l_3 and l_4 also become candidate polling points. In Fig. 1, if there is a wireless link between sensor s_i and position l_j, we say s_i belongs to the neighbor set of l_j, where $s_i \in \{s_1, s_2, s_3, s_4\}$ and $l_j \in \{l_1, l_2, \ldots, l_6\}$. Thus, the candidate polling point set $L = \{l_1, l_2, \ldots, l_6\}$; neighbor sets of l_1, l_2 and l_5 are $\{s_1, s_2\}$; neighbor sets of l_3, l_4 and l_6 are $\{s_3, s_4\}$. In summary, a candidate polling point set can contain two types of points in the plane: the positions where sensors are deployed and the points where the M-collector has tested the wireless links between it and its one-hop neighbors.

Unlike in a multi-hop homogeneous network, in our scheme M-collectors and sensors operate in a server-client mode, and sensors do not need to overhear the channel all the time to relay packets from their neighbors. Once an M-collector moves to a polling point, it can send a beep to wake up the transceiver of the nearby sensors, gather data from sensors, and put sensors into sleep again. We assume each sensor is equipped with a passive RFID device [15], which can activate the transceiver of the sensor by sending an interrupt signal to the micro-controller, once it receives the RFID call from the M-collector. The advantage of passive RFID devices is that they do not need power supply from sensor batteries and can obtain energy from outside RF signal. Next we will define the single hop data gathering problem.

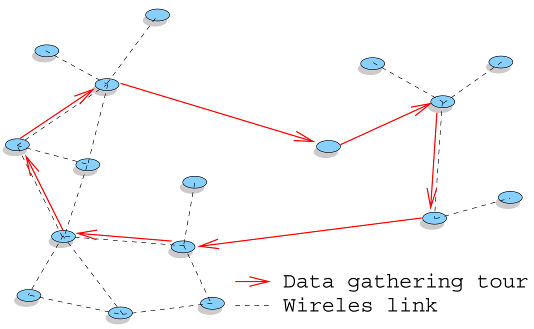


Fig. 2. Single hop data gathering problem.

III. Single-hop Data Gathering

In this section, we consider the problem of finding the shortest moving tour of an M-collector which visits the transmission range of each sensor. As shown in Fig. 2, the positions of sensors are either the polling points in the data gathering tour or within the one-hop range of the polling points. For the sake of simplicity, we assume M-collectors move at a fixed speed, and ignore the time for making turns and data transmission, such that we can roughly estimate the time consumption of a data gathering tour by the tour length. Clearly, by moving through the shortest tour, data can be collected in the shortest time such that the users will have the most up-to-date data. We refer to this problem as the single-hop data gathering problem, or SHDGP for short.

Before giving the formulation of the SHDGP problem, we first give some definitions and assumptions for the problem. Let R and r denote the radius of sensing field and transmission range of sensors, respectively. We consider the problem in which r and R must satisfy $0 < \frac{r}{R} < 1$. Note that when $\frac{r}{R} \to 0$, The M-collector must visit the location of every sensor one by one to gather data. In this case, the problem is reduced to the well-known *Traveling Salesman Problem (TSP) for points in the plane* [28]. The goal of *TSP for points in the plane* is to find a minimum distance (cost) tour that visits every point in the plane exactly once, which is known to be NP-hard. When $\frac{r}{R} \geq 1$, the network reduces to a single hop network, and like a base station in a WLAN, the M-collector can collect data directly from any sensor without moving. Considering the applications, such as environment monitoring and battlefield surveillance, in which sensors with fixed transmission power are deployed in a quite large area, we are interested in the more realistic case in which $0 < \frac{r}{R} < 1$.

After the candidate polling point set is obtained, the SHDGP problem can be formalized as follows. Given a set of sensor nodes $S = \{s_1, s_2, \ldots, s_{n_s}\}$, a set of candidate polling point set $L = \{l_0, l_1, l_2, \ldots, l_{n_l}\}$, where l_0 denotes the starting and the ending points of the tour, and the neighbor set $nb(l_i)$ of each candidate polling point l_i, $(i = 1, 2, \ldots, n_l)$, find a set of polling points and determine the sequence to visit them, such that every sensor in S belongs to the neighbor set of at least one polling point, and the total distance of line segments connecting all polling points is minimized. We define a complete direct graph $G = (L, A)$ and associate a non-negative cost c_{ij} with each arc $a_{ij} \in A$, where c_{ij} equals the cost of the distance between the candidate polling points l_i and l_j. The SHDGP problem may be formulated as a mixed integer program as follows:

$$\text{Minimize} \quad \sum_{i,j \in L, i \neq j} c_{ij} x_{ij} \tag{1}$$

Subject to

$$\sum_{i \in L, i \neq j} x_{ij} = I_j, \forall j \in L \tag{2}$$

$$\sum_{j \in L, j \neq i} x_{ij} = I_i, \forall i \in L \tag{3}$$

$$\sum_{j \in nb(l_i)} I_i \geq 1, \forall j \in S \tag{4}$$

$$y_{ij} \leq |L| x_{ij}, \forall i, j \in L \tag{5}$$

$$\sum_{j \in L \setminus \{l_0\}} y_{jl_0} = \sum_{j \in L \setminus \{l_0\}} I_j \tag{6}$$

$$\sum_i y_{ji} - \sum_k y_{kj} = I_j, \forall j \in L \setminus \{l_0\} \tag{7}$$

where

$$x_{ij} = \begin{cases} 1, & \text{if the data gathering tour contains arc } a_{ij} \\ 0, & \text{otherwise} \end{cases}$$

$$I_i = \begin{cases} 1, & \text{if the data gathering tour contains candidate polling point } l_i \\ 0, & \text{otherwise} \end{cases}$$

y_{ij} : the flow value from l_i to l_j on arc a_{ij}

In this formulation, the objective function (1) minimizes the total cost (distance) of the data gathering tour. x_{ij} is an indicator variable denoting whether arc a_{ij} from candidate polling points l_i to l_j belongs to the optimal tour. Binary variable I_i indicates whether candidate polling point l_i is on the optimal tour. Constraints (2) and (3) ensure the fact that every node in the tour must have one arc pointing towards it and the other arc pointing away from it. Constraint (4) enforces that every sensor must be in the neighbor set of at least one polling point belonging to the tour, such that every sensor can communicate with the M-collector directly. Constraints (5)-(7) exclude the solutions with sub-tours, which is similar to that in [23] proposed by Gavish for the capacitated minimal direct tree problem. Specifically, constraint (5) restricts that flow can only take place in an arc if it is on the tour. Constraint (6) specifies that the units of flow entering vertex l_0 equals the number of polling points in the tour. Finally, Constraint (7) enforces that one unit flows out of each of the other points in the tour. In a way similar to that in [23], it can be shown that constrains (5)-(7) can prohibit a tour which does not include the starting and ending point l_0.

We now show that the SHDGP problem is NP-hard. Consider a special case of the problem. When the transmission range of any sensor is close to 0, the M-collector must visit every sensor one by one to collect data, instead of simply entering their transmission ranges. Thus, the problem is reduced to finding the shortest tour of visiting every sensor exactly once, which is known to be another NP-hard problem, *Traveling Salesman Problem (TSP)*. Thus, SHDGP is NP-hard.

IV. HEURISTIC ALGORITHMS FOR THE SHDGP PROBLEM

We have known that the SHDGP problem is NP-hard. In this section, we will develop a heuristic algorithm for the problem. First, it is interesting to compare the SHDGP problem with a similar problem, the *Covering Salesman Problem (CSP)* [26]. In [26], Current and Schilling first defined the CSP and proved the NP-hardness of the CSP. The problem they considered is to obtain the shortest tour of a subset of all cities such that every city not on the tour is within some predetermined distance, $dist$, of a city that is on the tour. If the transmission range of each sensor could be modeled as a disk-shaped area, SHDGP can be simplified to the CSP problem by setting $dist$ in the CSP equal to the transmission range of sensors. In [26], a heuristic solution was introduced by dividing the problem into two NP-hard subproblems. First, find a minimum vertex cover and then determine the shortest tour of all vertices in the cover. The first step of their heuristic is to minimize the number of stops (same as polling points in SHDGP), but it does not consider the lengths of the edges connecting these stops.

Arkin and Hassin designed a $[(\sqrt{7^2+3^2}+1)r_{tsp}]$ (≈ 12.9) approximation algorithm for the geometric version of the CSP in [27], where the salesman has to enter the unit circle region around each city, and r_{tsp} denotes the best known constant factor we can approximate an optimal TSP tour on a set of points in the plane, and currently equals 1.5. The basic idea of their approximation algorithm is as follows. First, cover all unit circles by a minimum number of vertical covering lines. Then, choose a representing point from each unit circle region to be the intersection point of the diameter of the region and the covering line. Finally, find an approximation TSP tour on the representing points. We will refer to this approximation algorithm as *covering line algorithm* in the rest of the paper. Note that if the transmission range of each sensor node can be modeled as a unit circle, the covering line algorithm for the CSP could also approximate the optimal tour for SHDGP. However, as discussed earlier, in practice, transmission ranges of sensors are far from unit circles and difficult to be estimated. Furthermore, though the covering line algorithm provides an upper bound on the tour length, the approximation ratio is quite high (≈ 12.9). In Section VI, we will compare our proposed greedy algorithm with the covering line algorithm in various scenarios. In addition, we have managed to run the optimal algorithm for a few small networks and compare with our greedy algorithm. As will be seen, the simulation results show that our greedy algorithm performs much better than the covering line algorithm in various scenarios, and is close to the optimal algorithm in small networks.

The basic idea of our greedy algorithm is to choose a subset of points from the candidate polling point set, each of which corresponds to a neighbor set of sensors. At each stage of the algorithm, a neighbor set of sensors can be covered when its corresponding candidate polling point is chosen as a polling point in the data gathering tour. The algorithm will terminate after all sensors are covered. Using a similar idea in the greedy algorithms in [25] for the *Minimum Set Cover* problems, our greedy algorithm tries to cover each uncovered neighbor set of sensors with the minimum average cost at each stage, where the "cost" will be formally defined later. The algorithm can be described as follows. First, start with an empty set P. Let P contain all polling points. Let U contain the set of remaining uncovered sensors at each stage of the algorithm. Let F be the family of neighbor sets. Recall that each neighbor set corresponds to a candidate polling point and contains all the sensors that can be polled by the M-collector at this candidate polling point. The distance between two neighbor sets is defined as the distance between their corresponding candidate polling points. Let $cost\{S\}$ be the cost of an uncovered neighbor set S and equal to the shortest distance between S and any covered neighbor set. Let $\alpha = \frac{cost\{S\}}{|S \bigcap U|}$ and denote the average cost to cover each uncovered sensor in S. While there are remaining uncovered elements in U, choose the uncovered neighbor set S from F with the minimum α value, add the corresponding candidate polling point of S into P, cover all uncovered sensors in S, and remove them from U. The algorithm terminates when all nodes are covered. Finally, P contains all polling points in the data gathering tour.

After obtaining all the polling points, the data gathering tour can be easily obtained by running any approximate algorithm for the TSP problem. It is interesting to note that in a special case of SHDGP, when each neighbor set only contains one sensor and no two neighbor sets contain the same sensor. Our greedy algorithm is exactly same as Prim's algorithm for *Minimum Spanning Tree* problem, since the M-collector has to visit every candidate polling point to cover all sensors. Thus, we name our greedy algorithm as *Spanning Tree Covering* algorithm. An example of the spanning tree covering algorithm is illustrated in Fig. 3. Starting from the data sink, the M-collector chooses p_1 as the first polling point, since compared to other candidate polling points, the uncovered sensors in the neighbor set of p_1 can be covered with the smallest average cost $\frac{d_1}{3}$. Next, p_2 and p_3 will be picked as the second and the third polling points with the average cost $\frac{d_2}{3}$ and $\frac{d_3}{3}$, respectively. After that, the shortest tour can be approximated on all chosen polling points and the data sink, as shown in Fig. 3(d). The details of the spanning tree covering algorithm are given in Table 1.

We now analyze the communication and computational complexity of the spanning tree covering algorithm. Suppose there are a total of N sensors and M candidate polling points. Recall that a polling point can be either the position of a sensor or a point where the M-collector has tested the wireless links between it and its one-hop neighbors during the exploration phase. In a fully connected network where the connection pattern of sensors can be sent to the data sink via wireless transmissions before the M-collector starts the first tour, the M-collector does not have to explore the field, and the position of each sensor directly becomes a candidate polling point. In this case, the number of candidate polling points M equals the number of sensors N. If not all sensors are connected to the data sink, an M-collector needs to be sent out to discover the position of every sensor no matter which data gathering scheme will be used. This is because that in most applications, sensing data are related to the positions where they are collected to be meaningful. Thus, the procedure of obtaining candidate polling points does not cause much extra communication cost than the regular exploration procedure. In this case, the number of candidate polling points M is no less than the number of sensors N, and depends on several factors, such as the area of the sensing field A, the transmission range of sensors r and the exploration scheme. In a naive exploration scheme,

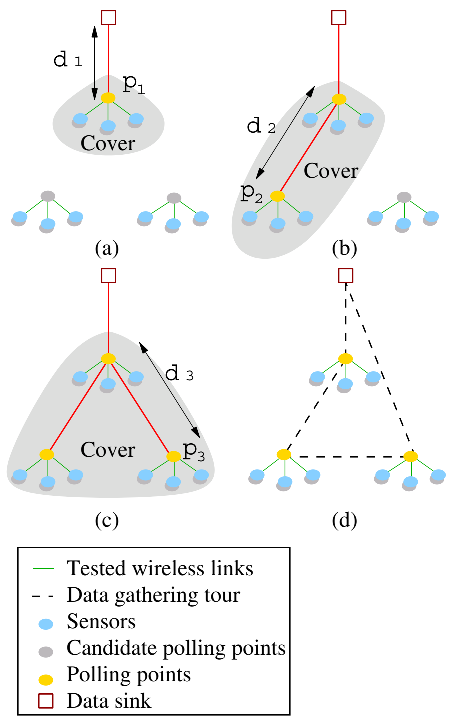


Fig. 3. Spanning tree covering algorithm. (a) The neighbor set of p_1 is covered with the average cost $\frac{d_1}{3}$. (b) The neighbor set of p_1 is covered with the average cost $\frac{d_2}{3}$. (c) The neighbor set of p_1 is covered with the average cost $\frac{d_3}{3}$. (d) The data gathering tour obtained by the spanning tree covering algorithm.

TABLE 1
SPANNING TREE COVERING ALGORITHM

```
**Spanning Tree Covering Algorithm**
Create an empty set $P$
Create a set $U$ containing all sensors
Create the family set $F$ of all neighbor sets
**while** $U \neq \Phi$
  Find the set $S \in F$ that minimizes $\alpha = \frac{cost\{S\}}{|S \bigcap U|}$
  Cover sensors in $S$
  Add the corresponding polling point of $S$ into $P$
  Remove sensors in $S$ from $U$
**end while**
Find an approximate shortest tour on polling points in $P$
```

for example, the M-collector moves along several parallel lines, M is in the order of $\frac{A}{r^2}$. Note that the exploration procedure is one-time operation and does not need to be repeated unless the topology of the network changes. For the computational complexity, it takes $O(NM)$ times to find a sub-family of neighbor sets which cover all sensors. The work of finding an approximate shortest tour on polling points in P can be done in $O(M^2)$ times. Thus, the total computational complexity of the spanning tree covering algorithm is $O(M^2 + NM)$.

V. DATA GATHERING WITH MULTIPLE M-COLLECTORS

We have considered how to plan the data gathering tour of a single M-collector. However, for some large-scale applications, each data gathering tour may take so long time that a single M-collector may not be enough to visit the transmission ranges of all sensors, before their buffers overflow. A possible solution [29] to this problem is to allow some sensors to relay packets from other nodes to the mobile data collector. Thus, the M-collector does not need to visit the transmission range of every single sensor, and the length of each tour can be reduced. However, the drawback of using relay is that the relaying nodes may fail faster than others. In order to avoid unbalanced network lifetime, we will stay with the one-hop data gathering scheme by utilizing multiple M-collectors.

A straightforward way to deal with the tour planning problem of multiple M-collectors can be borrowed from the traditional delivery vehicle routing problem, where a number of trucks are sent out from the facility center, each of them moves through a sub-tour, delivers packages home by home, and finally returns to the facility center before the center closes. The problem of finding a set of delivery sub-tours so that the number of trucks can be minimized is defined as *Multiple Traveling Salesman Problem (MTSP)* [30], which is known to be NP-hard. Different from delivery trucks, M-collectors can gather data remotely via wireless links without visiting the "home" of every sensor. Given the distance/time constraint of each sub-tour to prevent buffers from overflowing, the problem we consider is to find a set of data gathering sub-tours, such that the number of M-collectors can be minimized. If we force every M-collector to visit the data sink at least once in each tour like delivery trucks, the problem can be reduced to the MTSP problem by retracting the transmission range of any sensor to zero. Furthermore, in some very large scale networks, some sensors are deployed so far away from the data sink that the round-trip time consumption of the longest data gathering tour exceeds the time constraint for filling the buffer of a sensor. Thus, if all M-collectors have to visit the data sink, even there are enough number of M-collectors, a feasible set of sub-tours cannot always be found. Fortunately, unlike delivery vehicles, any two M-collectors can exchange their "goods" (data) via wireless links without meeting each other physically, which makes it possible for M-collectors far away from the data sink to upload data to the data sink through the relays of other M-collectors rather than visit the data sink themselves.

In our data gathering scheme with multiple M-collectors, only one of M-collectors needs to visit the transmission range of the data sink. As shown in Fig. 4 (b), (c) and (d), the entire network can be divided into some subnetworks. In each subnetwork, an M-collector is responsible to gather data from local sensors in the sub-area. Once a while, the M-collector forwards the sensing data to one of other nearby M-collectors, when two M-collectors move close enough. Finally data can be forwarded to the M-collector which will visit the data sink via relays of other M-collectors. In Fig. 4(d), all data are forwarded to M-collector 1 from other M-collectors, and then M-collector 1 carries and uploads data to the data sink. There are some interesting issues here, such as how to relay the packets to the data sink energy efficiently, how to schedule the movement of M-collectors to reduce the packet delay, and so on. In this paper, we will focus on how to plan the sub-tours of multiple M-collectors to minimize the number of M-collectors.

The data gathering algorithm with multiple M-collectors can be described as follows. First, find the polling point set P by running the spanning tree covering algorithm in Table 1. Then,

TABLE 2
DATA GATHERING ALGORITHM WITH MULTIPLE M-COLLECTORS

```
**Data Gathering Algorithm with Multiple M-collectors**
Find the polling point set $P$
Find the spanning covering tree $T$ on all polling points in $P$
For each vertex $v$ in $T$, calculate the weight value $Weight(v)$
while $T \neq \Phi$
  Find the deepest leaf vertex $u$ in $T$
  Let the root of the subtree $t$, $Root(t) = u$
  while $Weight(Parent(Root(t))) \leq \frac{L_{max}}{2}$
    $Root(t) = Parent(Root(t))$
  end while
  Add all child vertices of $Root(t)$ and edges connecting
    them into $t$ and remove $t$ from $T$
  Update weight value of each remaining vertex in $T$
end while
```

find the minimum spanning tree $T(V, E)$ on polling points. We refer to the minimum spanning tree on polling points as *Spanning Covering Tree*. Let L_{max} be the upper bound on the length of any sub-tour, which guarantees no buffer overflows. Let $t(v)$ denote the subtree of T, which is rooted at vertex v and consists of all child vertices of v and edges connecting them in T. Let $Parent\{v\}$ be the parent vertex of v in T. Let $Weight\{v\}$ represent the total length of the subtree $t(v)$ rooted at v. Then, calculate the weight values of all vertices in T. Repeatedly removing subtrees from T until no any vertex left in T. To build a subtree t in each loop, start from the deepest leaf vertex of the remaining T, and let it be the root $Root(t)$ of the subtree t. Check the weight of $Parent(Root(t))$, and let $Root(t) = Parent(Root(t))$, if $Weight(Parent(Root(t))) \leq \frac{L_{max}}{2}$. Otherwise, add all child vertices of $Root(t)$ and edges connecting them in T into t, and remove t from T. Here, $Weight(Parent(Root(t)))$ also denotes the total edge length of subtree t. After removing the subtree, upgrade the weight value of each vertex in the remaining T. The algorithm terminates when T is empty. Then T is decomposed into a set of subtrees. The total length of any subtree t, denoted by L_t, is no more than $\frac{L_{max}}{2}$. Finally, the sub-tour on polling points of each subtree can be determined by running the approximation algorithm for TSP. Let L^t_{apx} be the length of the approximated sub-tour on points in the subtree t. In the 2-approximation algorithm for TSP [28], the approximated tour is obtained by duplicating all edges of the minimum spanning tree and then finding an Eulerian circle in it. Hence, L^t_{apx} is no more than 2 times the length of the minimum spanning subtree t, L_t, that is, $L^t_{apx} \leq 2 \times L_t$. As discussed above, L_t is bounded by $\frac{L_{max}}{2}$. Thus, we have $L^t_{apx} \leq 2 \times L_t \leq L_{max}$, which means that the length of any sub-tour obtained by the data gathering algorithm with multiple M-collectors is no more than the upper bound on the length of a sub-tour L_{max}. The details of the algorithm are shown in Table 2. Fig. 4 (a)-(d) illustrate four major steps of the data gathering algorithm with multiple M-collectors: (1) Build the spanning covering tree. (2) Decompose the spanning covering tree into a set of subtrees. (3) Find an approximate shortest sub-tour on the points of each subtree. (4) Sensing data collected from sensors are forwarded to the nearest M-collector to the data sink.

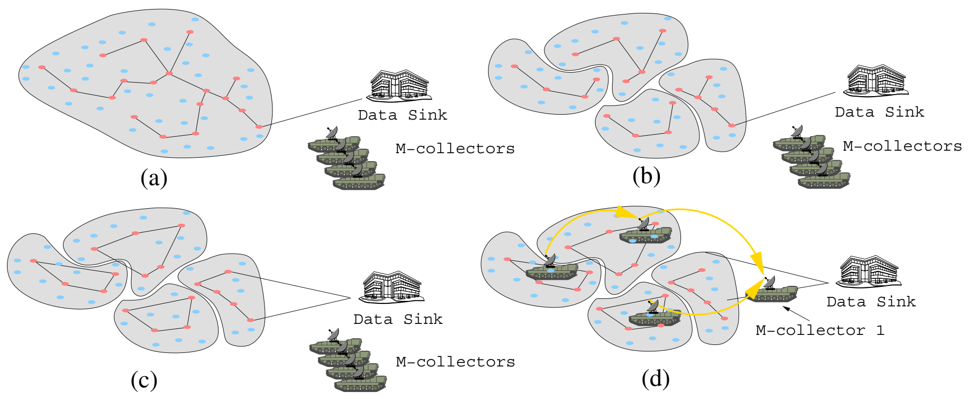


Fig. 4. Data gathering with multiple M-collectors. (a) Build the spanning covering tree. (b) Decompose the spanning covering tree into a set of subtrees. (c) Find an approximate shortest tour on points of each subtree. (d) Sensing data collected from sensors are forwarded to the nearest M-collector to the data sink.

We next discuss the complexity of the data gathering algorithm with multiple M-collectors. Compared to the spanning tree covering algorithm in Table 1, no extra communication is needed. In Table 2, both outer and inner while loops need to be executed at most $O(M)$ rounds, where M is the number of candidate polling points. The loop takes $O(M^2)$ time, and the operations before the loops take another $O(NM + M^2)$ time. Thus, the total computational complexity is $O(NM + M^2)$.

VI. Performance Evaluations

We have conducted extensive simulations to validate the algorithms we propose. In the simulations, we assume that a bunch of sensor nodes are uniformly deployed in the sensing field. Two-ray propagation model is used to describe the feature of the physical layer. The radio bandwidth is 200kbps. CBR traffic on the top of UDP is generated to measure the throughput. Each packet has a fixed size of 80 bytes, including header and payload. We evaluated the tour length of a single M-collector in both small and large networks, compared the relative network lifetime of the spanning tree covering algorithm with those of the other two data gathering schemes, and illustrated the data gathering algorithm with multiple M-collectors in a randomly generated network.

A. *Tour length of a single M-collector*

Because of the NP-hardness of the SHDGP problem, the brutal force search method of the optimal solution in a large network becomes impossible. However, we have managed to run the optimal algorithm for a few small networks for comparing with our heuristic algorithm. We first compare the tour length of a single M-collector obtained by running the spanning tree covering algorithm with the optimal solution in small networks. The mixed integer programming (MIP) formulation of SHDGP in Section III is solved to obtain the optimal solution by using GNU linear programming solver (GLPK) [31]. The connectivity of a sensor network depends on two major factors: deployment density and the transmission range of sensors. In this scenario, we measure the average tour length for all combinations of transmission range equal to $40m$, $60m$ and $80m$, and the number of nodes equal to 20, 30 and 40. Sensors are uniformly deployed in a $500m$ by $500m$ field. An M-collector starts each tour from point $(0m, 250m)$ and returns to the same point after each tour. In a very sparsely deployed network where the transmission range is relatively short compared to the average distance between a sensor and its nearest neighbor, it is very possible that the network is totally disconnected. In this case, the M-collector may need to visit every sensor, no matter which algorithm is employed. We can observe in Fig. 5 (a) that when 20 nodes with a relatively short transmission range $40m$ are deployed , both spanning tree covering algorithm and covering line algorithm have very close tour length to the optimal solution. As the transmission range increases, we can see in Fig. 5 (b) and (c), the spanning tree covering algorithm performs much better than the covering line algorithm and is very close to the optimal solution.

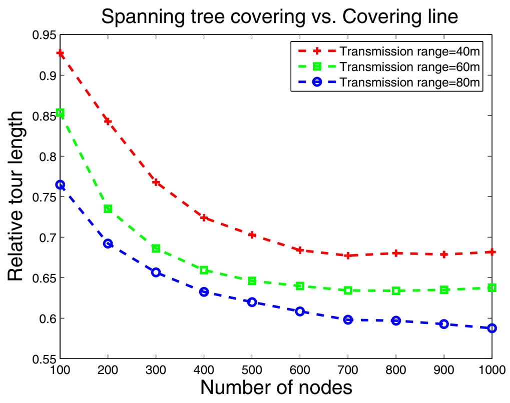
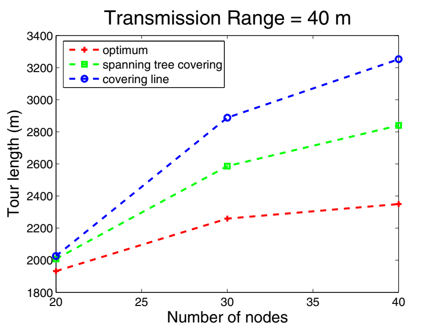
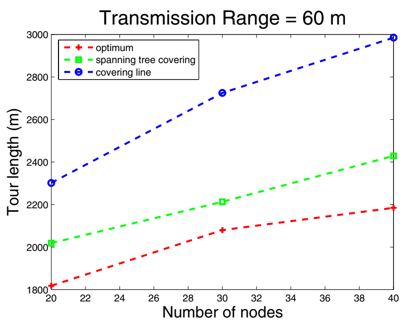


Fig. 6. Comparison with the covering line algorithm in large networks as the number of nodes increases from 100 to 1000.

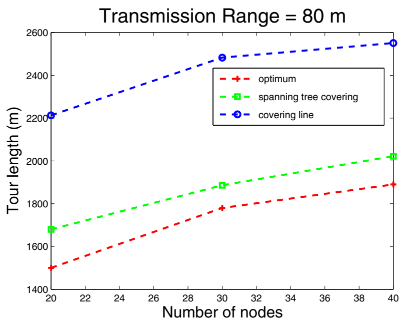
We also compare the tour length of the spanning tree covering algorithm with the covering line algorithm in larger networks. We measured the relative tour length of the spanning tree covering algorithm compared to that of the covering line algorithm for transmission ranges $40m$, $60m$ and $80m$, respectively, when the number of nodes increases from 100 to 1000. Sensors are randomly deployed into a $1000m$ by $1000m$ field. The M-collector needs to visit the data sink at point $(0m, 500m)$ in each tour. In Fig. 6, we can observe two facts: First, for the networks with the same size, the larger the transmission range is, the shorter average moving distance the M-collector needs to travel. Second, for any transmission range, the relative tour length keeps decreasing as the network size increases. For the network as large as 1000 nodes with the transmission range equal to $80m$, the spanning tree covering algorithm can save up to 40% moving distance compared to the covering line algorithm. In summary,



(a)



(b)



(c)

Fig. 5. Comparison with the optimal solution in small networks. (a) Transmission range of sensors equals 40m. (b) Transmission range of sensors equals 60m. (c) Transmission range of sensors equals 80m.

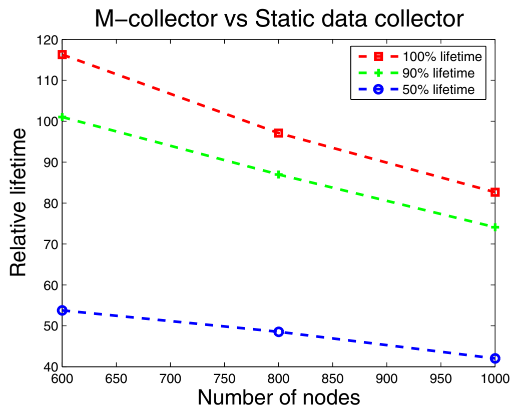
for a fixed sensing field, the higher deployment density and the longer transmission range lead to shorter moving distance of the M-collector.

B. Network lifetime

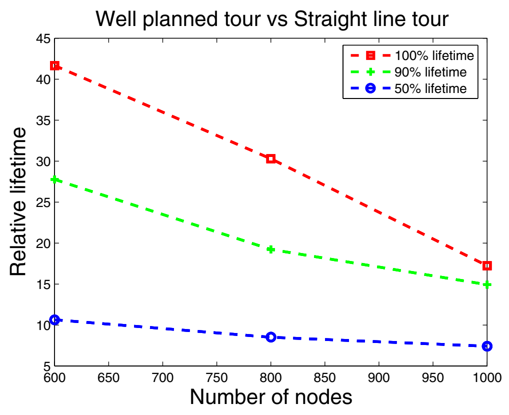
We now compare the network lifetime of the following three data gathering schemes: *Scheme 1*: A static data collector is placed in the center of the network (at point $(335m, 335m)$); *Scheme 2*: A mobile data collector which can only move back and forth through the straight line between $(0m, 335m)$ and $(670m, 335m)$; *Scheme 3*: An M-collector which can move along a well-planned data gathering tour that starts from and ends at point $(0m, 335m)$. In this scenario, we introduce a new metric, called *x% network lifetime*, which is defined as the network lifetime when $(100-x)\%$ sensors either run of battery or cannot send data to the data sink due to the failure of relaying nodes. In Schemes 1 and 2, sensors are allowed to relay packets. We measured the optimal network lifetime by using the load balancing algorithm in [19]. The optimal lifetime of the first two schemes is used as the performance reference for comparison purpose. In *Scheme 1*, since every sensor must be connected to the static data collector, sensors have to be deployed densely enough to guarantee the connectivity. We measured the 100% 90% and 50% network lifetimes of all three schemes, when 600, 800 and 1000 sensors are randomly deployed in a $670m$ by $670m$ area. The $x\%$ relative network lifetimes of *Scheme 3* compared to *Scheme 1* and *Scheme 2* are plotted in Fig. 7 (a) and (b). From Fig. 7, we observe that, by moving the M-collector through a well-planned tour, the lifetime of *Scheme 3* can be prolonged significantly compared to *Scheme 1* and *Scheme 2*.

C. Data gathering with multiple M-collectors

In this scenario, 600 sensors are deployed into $1000m \times 1000m$ area with a transmission range $40m$. Assume that M-collectors move at the fixed speed of $1m/s$. The data memory of each sensor can hold at most $512k$ bytes, while each sensor collects data from the environment at the fixed rate of $512Bps$. Thus, each sub-tour of any M-collector must be no more than $1000m$ to prevent buffers of sensors from overflowing. We show how the algorithm in Table 2 achieves sub-tour-planning for multiple M-collectors. Fig. 8(a), (b) and (c) give the simulation results for finding the spanning covering tree of the network, decomposing the spanning covering tree into a set of subtrees, and



(a)



(b)

Fig. 7. Relative x% network lifetime. (a) Comparison of M-collector and static data collector. (b) Comparison of well planned tour of M-collector and straight line tour.

approximating the shortest tour on points of each subtree, respectively, where the length of each tour is less than $1000m$.

VII. Conclusions

In this paper, we have proposed a new data gathering scheme for large scale sensor networks. We have introduced a mobile data collector, called M-collector, which works like a mobile base station in the network. An M-collector starts the data gathering tour periodically from the static data sink, traverses the entire sensor network, polls sensors and gathers the data from sensors one by one, and finally returns and uploads data to the

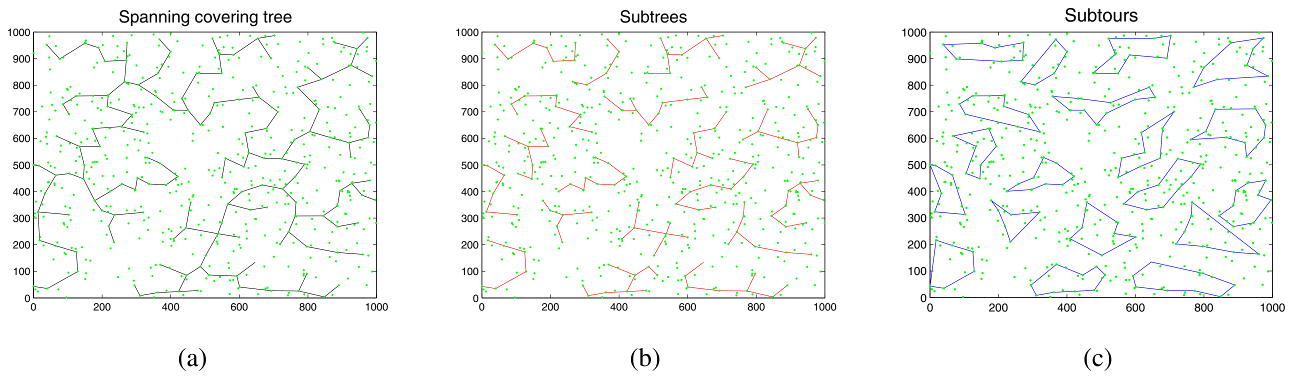


Fig. 8. Data gathering with multiple M-collectors. (a) The spanning covering tree. (b) The spanning covering tree is decomposed into a set of subtrees. (c) An approximate shortest sub-tour is found on nodes of each subtree.

data sink. Our single-hop data gathering scheme improves the scalability and solves intrinsic problems of large scale homogeneous networks. By introducing the M-collector, data gathering becomes more flexible and adaptable to the unexpected changes of the network topology. In addition, data gathering by M-collectors is perfectly suited for the applications, where sensors are only partially connected. For some applications in large scale networks with strict distance/time constraint for each data gathering tour, we have introduced multiple M-collectors by letting each of them move through a shorter sub-tour than the entire tour. The simulation results demonstrate that our greedy algorithm can greatly reduce the moving length compared to the covering line algorithm and is close to the optimal algorithm in small networks. Also, the proposed data gathering mechanism can prolong the network lifetime significantly compared to a network which has only a static data collector, or a network in which the mobile data collector can only move along straight lines.

References

[1] E. Biagioni and K. Bridges, "The application of remote sensor technology to assist the recovery of rare and endangered species," *International Journal of High Performance Computing Applications*, vol. 16, no. 3, Aug. 2002.

[2] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler and J. Anderson, "Wireless sensor networks for habitat monitoring," *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, Sept. 2002.

[3] S. Chessa and P. Santi, "Crash faults identification in wireless sensor networks," *Computer Communications*, vol. 25, no. 14, pp. 1273-1282, Sept. 2002

[4] L. Schwiebert, S. K. S. Gupta and J. Weinmann, "Research challenges in wireless networks of biomedical sensors," *ACM MobiCom 2001*, pp. 151-165.

[5] Networking the Hudson River, *http://www.technologyreview.com/Infotech/19309/page1/?a=f*, 2007.

[6] Wireless Bridge Monitoring, *http://www.dailywireless.org/2007/08/13/wireless-bridge-monitoring/*, 2007.

[7] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," in *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2002.

[8] T. Small and Z. Haas, "The shared wireless infostation model - a new ad hoc networking paradigm (or where there is a whale, there is a way)," *ACM MobiHoc 2003*.

[9] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin and P. Corke, "Data collection, storage and retrieval with an underwater sensor network," *Proc. of ACM Sensys*, 2005.

[10] A. Chakrabarty, A. Sabharwal and B. Aazhang, "Using predictable observer mobility for power efficient design of a sensor network," *Second International Workshop on Information Processing in Sensor Networks (IPSN)*, April 2003.

[11] B. Hull, V. Bychkovskiy, K. Chen, M. Goraczko, E. Shih, Y. Zhang, H. Balakrishnan and S. Madden, "CarTel: A distributed mobile sensor computing system," *Proceedings of SenSys*, 2006.

[12] A. Pentland, R. Fletcher and A. Hasson, "Daknet: rethinking connectivity in developing nations," *IEEE Computer*, vol. 37, no. 1, pp. 78-83, January 2004.

[13] D. Jea, A.A. Somasundara and M.B. Srivastava, "Multiple controlled mobile elements (data mules) for data collection in Sensor Networks," *2005 IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS '05)*, June 2005.

[14] A.A. Somasundara, A. Ramamoorthy and M.B. Srivastava, "Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines," *IEEE Real Time Systems Symposium (RTSS)*, December 2004.

[15] P. Skraba, H. Aghajan and A. Bahai, "RFID wakeup in event driven sensor networks," *SigComm04*, Portland, August 2004.

[16] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-efficient communication protocols for wireless microsensor networks," *Proc. of HICSS*, Maui, Hawai, Jan. 2000.

[17] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach," *IEEE INFOCOM 2004*, Hong Kong, 2004.

[18] Alan Amis, et. al, "Max-min D-cluster formation in wireless ad hoc networks," *IEEE INFOCOM 2000*, Tel-Aviv, Israel, 2000.

[19] Z. Zhang, M. Ma and Y. Yang, "Energy efficient multi-hop polling in clusters of two-layered heterogeneous sensor networks," *IEEE Transactions on Computers*, vol. 53, no. 2, pp. 231-245, Feb. 2008.

[20] A. Arora et al, "ExScal: Elements of an extreme scale wireless sensor network," *In Proc. 11th IEEE International Conference on Real-Time and Embedded Computing Systems and Applications (RTCSA '05)*, August 2005.

[21] G. Zhou, T. He, J. Stankovic and T. Abdelzaher, "RID: radio interference detection in wireless sensor networks," *IEEE INFOCOM 2005*, 2005.

[22] G. Zhou, T. He and J. Stankovic. "Impact of radio irregularity on wireless sensor networks," *In the Second International Conference on Mobile Systems, Applications, and Services* (MobiSys), June 2004.

[23] B. Gavish, "Formulations and algorithms for the capacitated minimal directed tree problem," *Journal of the ACM*, volume 30, 118–132, 1983.

[24] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, 1993.

[25] D. S. Johnson, "Approximation algorithms for combinatorial problems," *J. Comput. System Sci.* 9, 256-278, 1974.

[26] J. T. Current and D. A. Schilling, "The covering salesman problem," *Transportation Science*, 23:208– 213, 1989.

[27] E. Arkin and R. Hassin, "Approximation algorithms for the geometric covering salesman problem," *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 55, 1994.

[28] S. S. Skiena, "Traveling salesman problem," *The Algorithm Design Manual*, New York, Springer-Verlag, pp. 319-322, 1997.

[29] M. Ma and Y. Yang, "SenCar: An energy efficient data gathering mechanism for large-scale multihop sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 10, pp. 1476-1488, October 2007.

[30] G. N. Frederickson, M. S. Hecht and C. E. Kim, "Approximation algorithms for some routing problems," *SIAM J. Comp.* 7, 178-193, 1978.

[31] GNU Linear Programming Kit (GLPK) package, *http://www.gnu.org/software/glpk/*, 2007.