Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

ELSEVIER

Computer Networks 42 (2003) 697–716

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks ☆

Konstantinos Kalpakis *, Koustuv Dasgupta, Parag Namjoshi

Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA

Received 3 September 2002; accepted 26 December 2002

Responsible Editor: R. Sivakumar

Abstract

The rapid advances in processor, memory, and radio technology have enabled the development of distributed networks of small, inexpensive nodes that are capable of sensing, computation, and wireless communication. Sensor networks of the future are envisioned to revolutionize the paradigm of collecting and processing information in diverse environments. However, the severe energy constraints and limited computing resources of the sensors, present major challenges for such a vision to become a reality.

We consider a network of energy-constrained sensors that are deployed over a region. Each sensor periodically produces information as it monitors its vicinity. The basic operation in such a network is the systematic gathering and transmission of sensed data to a base station for further processing. During data gathering, sensors have the ability to perform in-network aggregation (fusion) of data packets enroute to the base station. The lifetime of such a sensor system is the time during which we can gather information from all the sensors to the base station. A key challenge in data gathering is to maximize the system lifetime, given the energy constraints of the sensors.

Given the location of n sensors and a base station together with the available energy at each sensor, we are interested in finding an efficient manner in which data should be collected from all the sensors and transmitted to the base station, such that the system lifetime is maximized. This is the maximum lifetime data gathering problem. In this paper, we describe novel algorithms, with worst-case running times polynomial in n, to solve the data gathering problem with aggregation in sensor networks. Our experimental results demonstrate that the proposed algorithms significantly outperform previous methods in terms of system lifetime.

Keywords: Sensor networks; Data gathering; Data aggregation; Energy-efficient protocols; Lifetime; Maximum flow; Minimum cut

1. Introduction

The recent advances in micro-sensor technology and low-power analog/digital electronics, have led to the development of distributed, wireless networks of sensor devices [11,18,19]. Sensor networks of the future are envisioned to consist of

☆ Supported in part by NASA under Cooperative Agreement NCC5-315 and Contract NAS5-32337, and by NSF under grant IRI-9729495.

* Corresponding author. Tel.: +1-410-455-3143; fax: +1-410-455-3969.

E-mail addresses: kalpakis@csee.umbc.edu (K. Kalpakis), dasgupta@csee.umbc.edu (K. Dasgupta), nam1@csee.umbc.edu (P. Namjoshi).

1389-1286/03/$ - see front matter
doi:10.1016/S1389-1286(03)00212-3

hundreds of inexpensive nodes, that can be readily deployed in physical environments to collect useful information (e.g. seismic, acoustic, medical and surveillance data) in a robust and autonomous manner. However, there are several obstacles that need to be overcome before this vision becomes a reality [9]. Such obstacles arise from the limited energy, computing capabilities and communication resources available to the sensors.

We consider a system of sensor nodes that are homogeneous and highly energy-constrained. Further, replenishing energy via replacing batteries on hundreds of nodes (in possibly harsh terrains) is infeasible. The basic operation in such a system is the systematic gathering of sensed data to be eventually transmitted to a base station for processing. The key challenge in such data gathering is conserving the sensor energies, so as to maximize their lifetime. To this end, there are several power-aware routing protocols for wireless ad hoc networks discussed in the literature [13,20]. In the context of sensor networks, LEACH [8] proposes a clustering-based protocol for transmitting data to the base station. The main features include local co-ordination for cluster formation among sensors, randomized rotation of cluster heads for improved energy utilization, and local data compression to reduce global communication. Chang and Tassiulas [3,4] describe data routing algorithms that maximize the time until the energies of the wireless nodes drain out. In related work, Bhardwaj et al. [2] derive upper bounds on the lifetime of a sensor network that collects data from a specified region using some energy-constrained nodes.

Data fusion or aggregation has emerged as a basic tenet in sensor networks. The key idea is to combine data from different sensors to eliminate redundant transmissions, and provide a rich, multi-dimensional view of the environment being monitored. Krishnamachari et al. [12] argue that this paradigm shifts the focus from address-centric approaches (finding routes between pairs of end nodes) to a more data-centric approach (finding routes from multiple sources to a destination that allows in-network consolidation of data). Madden et al. [8] describe the TinyOS operating system that can be used by an ad hoc network of sensors to locate each other and route data. The authors discuss the implementation of five basic database aggregates, i.e. COUNT, MIN, MAX, SUM, and AVERAGE, based on the TinyOS platform and demonstrate that such a generic approach for aggregation leads to significant power (energy) savings. The focus of the work in [17] is on a class of aggregation predicates that is particularly well suited to the in-network regime. Such aggregates can be expressed as an aggregate function f over the sets a and b, such that $f(a \cup b) = g(f(a), f(b))$. Other previous works [9,10,14,15] in the related area aim at reducing the energy expended by the sensors during the process of data gathering. Directed diffusion [10] is based on a network of nodes that can co-ordinate to perform distributed sensing of an environmental phenomenon. Such an approach achieves significant energy savings when intermediate nodes aggregate responses to queries. The SPIN protocol [9] uses meta-data negotiations between sensors to eliminate redundant data transmissions through the network. In PEGASIS [14], sensors form chains so that each node transmits and receives from a nearby neighbor. Gathered data moves from node to node, gets aggregated and is eventually transmitted to the base station. Nodes take turns to transmit so that the average energy spent by each node gets reduced. Lindsey et al. [15] describe a hierarchical scheme based on PEGASIS that reduces the average energy consumed and delay incurred in gathering the sensed data.

In this paper, we describe novel algorithms for data gathering and aggregation in sensor networks. We define the lifetime of a sensor network to be the time during which we can gather information from all the sensors to the base station. Given the location of sensors and the base station and the available energy at each sensor, we are interested in finding an efficient manner in which the data should be collected and aggregated from all the sensors and transmitted to the base station, such that the system lifetime is maximized. This is the maximum lifetime data aggregation (MLDA) problem. We first propose a near-optimal polynomial-time algorithm for solving the MLDA problem. The proposed algorithm, while performing significantly better than existing protocols

in terms of system lifetime, is computationally expensive for large sensor networks. Consequently, we describe clustering-based heuristics approaches for maximum lifetime data gathering and aggregation in large-scale sensor networks. Finally, we provide experimental results to show that (i) for smaller sensor networks the MLDA algorithm achieves system lifetimes that are 1.15–2.32 times better when compared to an existing data gathering protocol, (ii) for larger networks, our clustering-based heuristics achieve as much as a factor of 2.61 increase in the system lifetime when compared to the same protocol.

The rest of the paper is organized as follows. In Section 2, we describe the system model and the data gathering problem. Section 3 provides a detailed description of our MLDA algorithm to solve the maximum lifetime data aggregation problem in sensor networks. Next, in Section 4, we propose clustering-based heuristics to solve the problem efficiently for larger sensor networks. In Section 5, we present the experimental results and finally we conclude the paper in Section 6.

2. The data gathering problem

2.1. System model

Consider a network of n sensor nodes $1, 2, \ldots, n$ and a base station node t labeled $n+1$ distributed over a region. The locations of the sensors and the base station are fixed and known a priori. Each sensor produces some information as it monitors its vicinity. We assume that each sensor generates one data packet per time unit to be transmitted to the base station. For simplicity, we refer to each time unit as a *round*. We assume that all data packets have size k bits. The information from all the sensors needs to be gathered at each round and sent to the base station for processing. We assume that each sensor has the ability to transmit its packet to any other sensor in the network or directly to the base station. Further, each sensor i has a battery with finite, non-replenishable energy $\mathscr{E}_i$. Whenever a sensor transmits or receives a data packet, it consumes some energy from its battery. The base station has an unlimited amount of energy available to it.

Our energy model for the sensors is based on the *first order radio model* described in [8]. A sensor consumes $\epsilon_{\text{elec}} = 50$ nJ/bit to run the transmitter or receiver circuitry and $\epsilon_{\text{amp}} = 100$ pJ/bit/m^2 for the transmitter amplifier. Thus, the energy consumed by a sensor i in receiving a k-bit data packet is given by,

$$RX_i = \epsilon_{\text{elec}} k \tag{1}$$

while the energy consumed in transmitting a data packet to sensor j is given by

$$TX_{i,j} = \epsilon_{\text{elec}} k + \epsilon_{\text{amp}} d_{i,j}^2 k, \tag{2}$$

where $d_{i,j}$ is the distance between nodes i and j.

2.2. Problem statement

We define the *lifetime* T of the system to be the number of rounds until the first sensor is drained of its energy. A *data gathering schedule* specifies, for each round, how the data packets from all the sensors are collected and transmitted to the base station. For brevity, we refer to a data gathering schedule simply as a schedule. Observe that a schedule can be thought of as a collection of T directed trees, each rooted at the base station and spanning all the sensors i.e. a schedule has one tree for each round. The lifetime of a schedule equals the lifetime of the system under that schedule. Clearly, the system lifetime is intrinsically connected to the data gathering schedule. Our objective is to find a schedule that maximizes the system lifetime T.

3. MLDA: maximum lifetime data gathering with aggregation

Data aggregation performs in-network fusion of data packets, coming from different sensors enroute to the base station, in an attempt to minimize the number and size of data transmissions and thus save sensor energies [8,10,12,14]. Such aggregation can be performed when the data from different sensors are highly correlated. As in previous work [8,10,12,14], we make the simplistic

assumption that an intermediate sensor can aggregate multiple incoming packets into a single outgoing packet.

The MLDA problem. Given a collection of sensors and a base station, together with their locations and the energy of each sensor, find a data gathering schedule with maximum lifetime, where sensors are permitted to aggregate incoming data packets.

Consider a schedule $\mathscr{S}$ with lifetime T rounds. Let $f_{i,j}$ be the total number of packets that node i (a sensor) transmits to node j (a sensor or base station) in $\mathscr{S}$. Since any valid schedule must respect the energy constraints at each sensor, it follows that for each sensor $i = 1, 2, \ldots, n$,

$$\sum_{j=1}^{n+1} f_{i,j} TX_{i,j} + \sum_{j=1}^{n} f_{j,i} RX_i \leqslant \mathscr{E}_i. \tag{3}$$

Recall that each sensor, for each one of the T rounds, generates one data packet that needs to be collected, possibly aggregated, and eventually transmitted to the base station.

The schedule $\mathscr{S}$ induces a flow network $G = (V, E)$. The flow network G is a directed graph having as nodes all the sensors and the base station, and having edges (i, j) with capacity $f_{i,j}$ whenever $f_{i,j} > 0$.

Theorem 1. *Let $\mathscr{S}$ be a schedule with lifetime T, and let G be the flow network induced by $\mathscr{S}$. Then, for each sensor s, the maximum flow from s to the base station t in G is $\geqslant T$.*

Proof. Each data packet transmitted from a sensor must reach the base station. Observe that, the packets from s could possibly be aggregated with one or more packets from other sensors in the network. Intuitively, we need to guarantee that each of the T values from s *influences* the final value(s) received at the base station. In terms of network flows, this implies that sensor s must have a maximum $s - t$ flow of size $\geqslant T$ to the base station in the flow network G. □

Thus, a necessary condition for a schedule to have lifetime T is that each node in the induced flow network can push flow T to the base station t. Stated otherwise, each sensor s must have a minimum $s - t$ cut of capacity (size) $\geqslant T$ to the base station [5]. Next, we consider the problem of finding a flow network G with maximum T, that allows each sensor to push flow T to the base station, while respecting the energy constraints in (5) at all the sensors. We call such a flow network G an *admissible* flow network with lifetime T. An admissible flow network with maximum lifetime is called an *optimal admissible* flow network. Clearly, what needs to be found are the capacities of the edges in G.

3.1. Finding a near-optimal admissible flow network

An optimal admissible flow network can be found using an integer program with linear constraints. The integer program, in addition to the variables for the lifetime T and the edge capacities $f_{i,j}$, uses the following variables: for each sensor $k = 1, 2, \ldots, n$, let $\pi_{i,j}^{(k)}$ be a flow variable indicating the flow that k sends to the base station t over the edge (i, j).

The integer program computes the maximum system lifetime T subject to the energy constraint (5) and the additional linear constraints (6)–(9) for each sensor, as shown in Table 1. For each sensor $k = 1, 2, \ldots, n$, constraints (6) and (7) enforce the flow conservation principle at the sensor; constraint (9) ensures that T flow from sensor k reaches the base station; and constraint (8) ensures that the capacity constraints on the edges of the flow network are respected. Moreover, constraint (5) is used to guarantee that the edge capacities of the flow network respect the sensor's available energy. Finally, for the integer program, all variables are required to take non-negative integer values. The linear relaxation of the above integer program, i.e. when all the variables are allowed to take fractional values, can be computed in polynomial-time. Then, we can obtain a very good approximation for the optimal admissible flow network by first fixing the edge capacities to the floor of their values obtained from the linear relaxation so that the energy constraints are all satisfied; and then solving the linear program (4) subject to constraints (6)–(9) without requiring

Table 1
Integer program for finding an optimal admissible flow network for the MLDA problem

Objective:

$$\text{maximize } T \tag{4}$$

Constraints:

$$\sum_{j=1}^{n+1} f_{i,j} TX_{i,j} + \sum_{j=1}^{n} f_{j,i} RX_i \leqslant \mathscr{E}_i, \tag{5}$$

$$\sum_{j=1}^{n} \pi_{j,i}^{(k)} = \sum_{j=1}^{n+1} \pi_{i,j}^{(k)}, \quad \forall i = 1,2,\ldots,n \quad \text{and} \quad i \neq k, \tag{6}$$

$$T + \sum_{j=1}^{n} \pi_{j,k}^{(k)} = \sum_{j=1}^{n+1} \pi_{k,j}^{(k)}, \tag{7}$$

$$0 \leqslant \pi_{i,j}^{(k)} \leqslant f_{i,j}, \quad \forall i = 1,2,\ldots,n \quad \text{and} \quad \forall j = 1,2,\ldots,n+1, \tag{8}$$

$$\sum_{i=1}^{n} \pi_{i,n+1}^{(k)} = T, \tag{9}$$

where $k = 1,\ldots,n$ and all variables are required to be non-negative integers.

anymore that the flows are integers (since a solution with integer flows can always be found). [1]

3.2. Constructing a schedule from an admissible flow network

Next, we discuss how to get a schedule from an admissible flow network. Recall that a schedule is a collection of directed trees that span all the sensors and the base station, with one such tree for each round. Each such tree specifies how data packets are gathered and transmitted to the base station. We call these trees *aggregation trees*. An aggregation tree may be used for one or more rounds; we indicate the number of rounds f, that an aggregation tree is used, by associating the value f with each one of its edges; we call f to be the lifetime of the aggregation tree.

Fig. 1 provides an example of an admissible flow network G with lifetime $T = 100$ and two aggregation trees A_1 and A_2, with lifetimes 60 and 40 rounds respectively. By looking at one of these trees, say A_1, we see that for each one of 60 rounds, sensors 2 and 3 transmit one data packet to sensor 1, which in turn aggregates the incoming packets with its own data packet, and then sends one data packet to the base station. Similarly, for each of the remaining 40 rounds (using A_2), sensors 1 and 2 transmit one data packet to sensor 3, which in aggregates the incoming packets with its own packet, and sends one data packet to the base station. We next describe an algorithm to construct aggregation trees from an admissible flow network G with lifetime T.

Definition 1. Given an admissible flow network G with lifetime T and a directed tree A rooted at the base station t with lifetime f, we define the (A,f)-*reduction* G' *of* G to be the flow network that results from G after reducing by f, the capacities of all of its edges that are also in A. We call G' the (A,f)-*reduced* G.

Definition 2. An (A,f)-reduction G' of G is *feasible* if the maximum flow from v to the base station t in G' is $\geqslant T - f$ for each vertex v in G'.

[1] The reduction in the system lifetime achieved, w.r.t. the fractional optimal lifetime, is at most the maximum cardinality of any min $s-t$ cut.

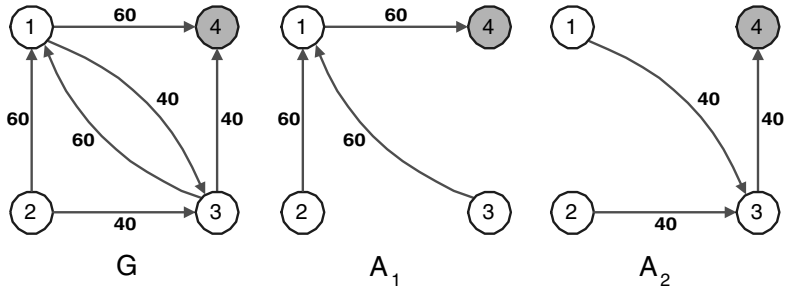


Fig. 1. An admissible flow network G with lifetime 100 rounds, and two aggregation trees A_1 and A_2 with lifetimes 60 and 40 rounds respectively.

Note that A does not have to span all the vertices of G, and thus it is not necessarily an aggregation tree. Moreover, if A is an aggregation tree, with lifetime f, for an admissible flow network G with lifetime T, and the (A,f)-reduction of G is feasible, then the (A,f)-reduced flow network G' of G is an admissible flow network with lifetime $T-f$. Therefore, we can devise a simple iterative procedure, to construct a schedule for an admissible flow network G with lifetime T, provided we can find such an aggregation tree A.

We use the GETTREE algorithm in Fig. 2 to get an aggregation tree A with lifetime $f \leqslant T$ from an admissible flow network G with lifetime T. Throughout this algorithm, we maintain the invariant that A is a tree rooted at t and the (A,f)-reduction of G is feasible. Tree A is formed as follows. Initially A contains just the base station. While A does not span all the sensors, we find and add to A an edge $e=(i,j)$, where $i \notin A$ and $j \in A$, provided that the (A',f)-reduction of G is feasible—here A' is the tree A together with the edge e, and f is 1 *or* the minimum of the capacities of the edges in A'.

Finally, we can compute a collection of aggregation trees from an admissible flow network G with lifetime T by using the GETSCHEDULE algorithm in Fig. 3, such that T data packets from each of the sensors are aggregated and transmitted to the base station t.

We refer to the algorithm described in this section, for computing a maximum lifetime data gathering schedule, as the MLDA algorithm.

Theorem 2. *Let $G=(V,E)$ be an admissible flow network with edge capacities $c(e)$, $e \in E$. Let $t \in V$ be the base station and T a positive integer i.e. the lifetime of the system. Then, the* GETSCHEDULE *algorithm can always find a sequence of aggregation trees that can be used to aggregate and transmit T data packets from each sensor to the base station.*

```
GETTREE (Flow Network G, Lifetime T, Base Station t)
1   initialize f ← 1
2   let A = (V_o, E_o) where V_o = {t} and E_o = ∅
3   while A does not span all the nodes of G do
4       for each edge e = (i, j) ∈ G such that i ∉ V_o and j ∈ V_o do
5           let A' be A together with the edge e
6           // check if the (A', 1)–reduction of G is feasible
7           let G_r be the (A', 1)–reduction of G
8           if MAXFLOW(v, t, G_r) ≥ T − 1 for all nodes v of G
9               // replace A with A'
10              V_o ← V_o ∪ {i}, E_o ← E_o ∪ {e}
11              break
12  let c_min be the minimum capacity of the edges in A
13  let G_r be the (A, c_min)–reduction of G
14  if MAXFLOW(v, t, G_r) ≥ T − c_min for all nodes v of G
15      f ← c_min
16  replace G with the (A, f)–reduction of G
17  return f, G, A
```

Fig. 2. Constructing aggregation tree A with lifetime f from an admissible flow network G with lifetime T.

```
GETSCHEDULE(Flow Network G, Lifetime T, BaseStation t)
1   S ← ∅
2   while T > 0 do
3       [f, G, A] ← GETTREE(G, T, t)
4       S ← S ∪ {A}
5       T ← T − f
6   return S
```

Fig. 3. Constructing a schedule $\mathcal{S}$ from an admissible flow network G with lifetime T.

Proof. Refer to Appendix A for proof details. □

3.3. Worst-case running time of MLDA

The running time analysis of the MLDA algorithm is based on the following three lemmas.

Lemma 1 (Bertsimas and Tsitsiklis [1]). *An ϵ-optimal solution to a linear program with n variables can be found in $O(n^{3.5}\log(1/\epsilon)+n^5\log(nU))$ time, where U is the maximum magnitude of the coefficients of the linear program.*

Proof. See the complexity analysis of the potential reduction algorithm in [1, p. 418]. □

Lemma 2 (Goldberg and Rao [7]). *Given a flow network $G=(V,E)$ with integral edge capacities bounded by U, a maximum $s-t$ flow can be computed in $O(\min(V^{2/3},E^{1/2})E\log(V^2/E)\log U)$ time, where s and t are any two vertices of G.*

Proof. See [7]. □

Lemma 3. *Consider a sensor network with a base station t and n sensors. Each sensor has initial energy no more than $\mathscr{E}_{\max}$ and consumes energy for receiving and transmitting k-bit data packets according to the first order radio model* [8]. *Then, the lifetime of the sensor network is $\leqslant \mathscr{E}_{\max}/(\epsilon_{\text{elec}}k)=O(1)$.*

Proof. Let $d_{\min}$ be the minimum distance of a sensor from the base station t. In each round, one data packet needs to be collected from each of the sensors and transmitted to the base station. From the first order radio model, Eqs. (1) and (2), the minimum total energy expended by all the sensors in one round is at least

$$\begin{aligned}&(n-1)\epsilon_{\text{elec}}k+(\epsilon_{\text{elec}}k+\epsilon_{\text{amp}}d_{\min}^2k)\\&\quad=n\epsilon_{\text{elec}}k+\epsilon_{\text{amp}}d_{\min}^2k,\end{aligned}\tag{10}$$

since there are $n-1$ packet transmissions over distance at least 0, and one packet transmission over distance at least $d_{\min}$. Therefore, the system lifetime T, i.e. the number of rounds before the first sensor is drained of its energy, is at most

$$\begin{aligned}&\frac{\text{total energy of all the sensors}}{\text{minimum energy consumed in one round}}\\&=\frac{n\mathscr{E}_{\max}}{n\epsilon_{\text{elec}}k+\epsilon_{\text{amp}}d_{\min}^2k}\leqslant\frac{\mathscr{E}_{\max}}{\epsilon_{\text{elec}}k}=O(1).\end{aligned}\tag{11}$$

□

Theorem 3. *The worst-case running time of the MLDA algorithm is $O(n^{15}\log n)$, where n is the number of sensors.*

Proof. Consider a sensor network with n sensors and maximum lifetime T.

To compute an admissible flow network, the MLDA algorithm solves the linear program (4) twice. Observe that the linear program (4) has $O(n^3)$ variables and coefficients of magnitude $O(T)$. Using Lemmas 3 and 1, it follows that the time t_{LP} to compute an ϵ-approximate solution to the linear program (4) is

$$t_{\text{LP}}=O\left(n^{10.5}\log\frac{1}{\epsilon}+n^{15}\log n\right)=O(n^{15}\log n),\tag{12}$$

since ϵ is assumed to be a small constant.

Next, we look at the running time of the GETSCHEDULE procedure. The GETSCHEDULE procedure makes $O(T)$ calls to the GETTREE routine. Each call of the GETTREE routine involves $O(V^2E)$ MAXFLOW computations on the admissible flow network $G=(V,E)$. [2] Since G has $O(n)$ vertices and $O(n^2)$ edges with capacity $O(T)=O(1)$, using Lemma 2, the running time of a MAXFLOW computation on G is $t_{\text{MAXFLOW}}=O(n^{8/3}\log n)$. Thus, the running time of the GETSCHEDULE routine is

$$\leqslant TV^2Et_{\text{MAXFLOW}}=O(n^{20/3}\log n).\tag{13}$$

Therefore, the total worst-case running time of MLDA is

[2] Note that since each edge of G needs to be checked at most once for inclusion in the aggregation tree [16], the GETTREE routine can be implemented to use $O(VE)$ MAXFLOW computations.

$$\leqslant 2t_{\mathrm{LP}} + \mathrm{O}(n^{20/3}\log n)$$
$$= \mathrm{O}(n^{15}\log n) + \mathrm{O}(n^{20/3}\log n)$$
$$= \mathrm{O}(n^{15}\log n). \quad (14)$$

We report actual running times of sample runs of the MLDA algorithm in Section 5. □

4. CMLDA: clustering-based MLDA

Given the location of n sensors and a base station t, we can find a maximum lifetime data gathering schedule using the MLDA algorithm. However, it involves solving a linear program (in Table 1) with $\mathrm{O}(n^3)$ variables and constraints. For large sensor networks, i.e. for large values of n, this can be computationally expensive.

In this section, we propose a clustering-based approach to solve the data gathering problem efficiently for large networks. In particular, we describe two heuristics—GREEDY CMLDA and INCREMENTAL CMLDA, based on the MLDA algorithm in Section 3.

Consider the set of n sensors $1, 2, \ldots, n$ and a base station t labeled as $n+1$. We assume, without loss of generality, that each sensor has initial energy $\mathscr{E}$. Let the sensors be partitioned into m clusters $\phi_1, \ldots, \phi_m$ each consisting of at most c sensors, i.e. $|\phi_i| \leqslant c$, for $i = 1, 2, \ldots, m$, where c is some constant. We refer to each cluster as a *super-sensor*. Such a partitioning of the sensors can be achieved using an appropriate proximity-based clustering algorithm. Let super-sensor ϕ_{m+1} consist only of the base station t.

4.1. The greedy CMLDA heuristic

We propose a simple heuristic for finding a maximum lifetime data gathering schedule in sensor networks. Our approach is to compute a maximum lifetime schedule for the super-sensors $\phi_1, \ldots, \phi_m$ with the base station ϕ_{m+1}, and then use this schedule to construct aggregation trees for the sensors.

Fig. 6 gives a high level view of the greedy clustering-based MLDA (GREEDY CMLDA) heuristic. In the first phase, we assign the initial energy of each super-sensor ϕ_i $(i = 1, 2, \ldots, m)$ to be the sum of the initial energies of the sensors within it, i.e. $\mathscr{E}_{\phi_i} = \mathscr{E}|\phi_i|$. The initial energy of super-sensor ϕ_{m+1} is set to infinity. The distance between two super-sensors ϕ_i and ϕ_j is assigned to be the maximum distance between any two nodes (sensor or base station) u and v, such that $u \in \phi_i$ and $v \in \phi_j$. Having set up the initial energies and the distances between the super-sensors, we can find a maximum lifetime schedule for the super-sensors $\phi_1, \ldots, \phi_m$ with the base station as ϕ_{m+1}, using the MLDA algorithm. Recall that such a schedule consists of a collection of directed trees $\mathscr{T}_1, \ldots, \mathscr{T}_k$, each rooted at ϕ_{m+1} and spanning over all the super-sensors. To distinguish it from an aggregation tree for the sensors, we refer to each such tree as an aggregation super-tree (or simply an *AS-tree*).

Next, we use the BUILD-TREE procedure (in Fig. 4) to construct an aggregation tree A for the sensors from an *AS*-tree $\mathscr{T}_k$. Observe that A is a directed tree rooted at t that is used to aggregate one data packet from each sensor. We denote $\mathscr{E}^r[i]$ to be the residual energy at sensor i. Initially, $\mathscr{E}^r[i] = \mathscr{E}$ for each sensor i in the network. Our objective is to construct (one or more) aggregation trees such that the *minimum residual energy* among the n sensors is maximized, thereby maximizing the lifetime of the corresponding data gathering schedule.

Initially, aggregation tree A contains only the base station t. We perform a (pre-order) traversal

BUILD–TREE(AS–Tree $\mathscr{T}_k$, Super–Sensor ϕ, Aggregation Tree A, Base Station t)
1 while A does not contain all the sensors in ϕ do
2 find a pair (i, j), where $i \in \phi - A$ and $j \in A$, with *maximum residual energy*
3 add the edge (i, j) to A
4 update the residual energy of sensor i as $\mathcal{E}^r[i] \leftarrow \mathcal{E}^r[i] - T\mathrm{x}_{i,j}$
5 if $j \neq t$ then update the residual energy of j as $\mathcal{E}^r[j] \leftarrow \mathcal{E}^r[j] - R\mathrm{x}_j$
6 foreach child ϕ' of ϕ in $\mathscr{T}_k$ do
7 $A \leftarrow$ BUILD–TREE$(\mathscr{T}_k, \phi', A, t)$
8 return A

Fig. 4. Constructing an aggregation tree A for the sensors from an *AS*-tree $\mathscr{T}$.

of the *AS*-tree $\mathscr{T}_k$. For each visited super-sensor ϕ, we add the sensors in ϕ to the current aggregation tree A. Let $\phi - A$ denote the set of sensors in ϕ that are not included in A. We define the *residual energy* of a pair (i,j) as $\min\{\mathscr{E}^r[i] - TX_{i,j}, \mathscr{E}^r[j] - RX_j\}$, where $i \in \phi - A$ and $j \in A$. Intuitively, on adding a directed edge (i,j) to A, the residual energy at sensor i is reduced by the energy consumed in transmitting a data packet from i to j. Moreover, if j is not the base station, its residual energy is reduced by the energy consumed in receiving a data packet. Among all pairs (i,j), such that $i \in \phi - A$ and $j \in A$, the BUILD-TREE procedure chooses one with the maximum residual energy and includes the edge (i,j) in A. The process is repeated until all sensors in ϕ are included in A, upon which it continues with the next super-sensor in $\mathscr{T}_k$. Fig. 5 gives an illustration of the BUILD-TREE procedure. The running time of the procedure is polynomial in the number of sensors. Finally, observe that a maximum lifetime schedule for the super-sensors could possibly consist of one or more *AS*-trees. In this case, we choose (in step 9 of Fig. 6) the *AS*-trees in decreasing order of their respective lifetimes; while constructing no more than f_k aggregation trees from a particular *AS*-tree $\mathscr{T}_k$, where f_k is the lifetime of the *AS*-tree $\mathscr{T}_k$.

Worst-case running time of GREEDY CMLDA. Consider the GREEDY CMLDA heuristic in Fig. 6. Steps 1 and 2 (of Phase I) use a greedy proximity-based algorithm to cluster the n sensors into $m+1$ super-sensors, which can be done in $O(n^2)$ time. In steps 3 and 4, we initialize the energies of and distances between the super-sensors in $O(n^2)$ time. Then, we compute an admissible flow network and a schedule for the super-sensors in $O(m^{15}\log m)$ worst-case running time, by solving the linear program (4) with $O(m^3)$ variables (see also Lemma 1 and Theorem 3).

In Phase II of the heuristic, we use this flow network to construct aggregation trees for the sensors. In steps 6 and 7, we initialize the lifetime and the residual energy of each sensor in $O(n)$ time. The *while* loop in step 8 is executed at most $T = O(1)$ times, where T is the lifetime found in Phase I. The time for each iteration of this *while*

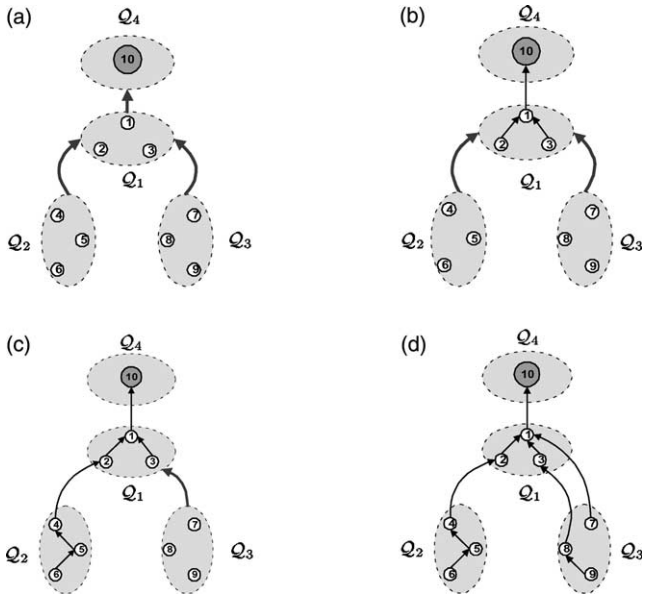


Fig. 5. Illustration of the BUILD-TREE procedure for an *AS*-tree $\mathscr{T}_k$. Sensors $1,2,\ldots,9$ are partitioned among super-sensors Q_1, Q_2, and Q_3. Super-sensor Q_4 contains the base station 10. The aggregation tree A is used for collecting one data packet from each sensor and transmitting to the base station. (a) *AS*-tree $\mathscr{T}_k$ for super-sensors Q_1, Q_2 and Q_3 with base station 10 in Q_4, (b) BUILD-TREE$(\mathscr{T}_k, Q_1, A, 10)$, (c) BUILD-TREE$(\mathscr{T}_k, Q_2, A, 10)$ and (d) BUILD-TREE$(\mathscr{T}_k, Q_3, A, 10)$.

INPUT:
Location of n sensors $1, 2, \ldots, n$ and a base station t.
Initial energy $\mathcal{E}$ in each sensor.

OUTPUT:
A data gathering schedule $\mathcal{S}$, i.e. a collection of aggregation trees, with lifetime T.

ALGORITHM:

PHASE I:

1. partition the sensors into m super–sensors $\phi_1, \ldots, \phi_m$
2. let super–sensor ϕ_{m+1} consist only of the base station t
3. let the energy of each super–sensor ϕ_i, $i = 1, 2, \ldots, m$, be $\mathcal{E}_{\phi_i} \leftarrow \mathcal{E} \cdot |\phi_i|$
4. let the distance between any two super–sensors ϕ_i and ϕ_j be $d_{\phi_i,\phi_j} \leftarrow \max\{d_{u,v} : u \in \phi_i, v \in \phi_j\}$
5. find an admissible flow network G for the super–sensors $\phi_1, \ldots, \phi_m$ with base station ϕ_{m+1}, and compute a schedule $\mathcal{T} \leftarrow \{\mathcal{T}_i, \ldots, \mathcal{T}_k\}$ from G

PHASE II:

6. initialize the schedule $\mathcal{S} \leftarrow \emptyset$ and lifetime $T \leftarrow 0$
7. let the residual energy of each sensor $i = 1, 2, \ldots, n$ be $\mathcal{E}^r[i] = \mathcal{E}$
8. while $\min\{\mathcal{E}^r[i] : i = 1, 2, \ldots, n\} > 0$ do
9. choose an AS–tree $\mathcal{T}_k$ from $\mathcal{T}$
10. initialize A to contain only the base station t
11. compute an aggregation tree $A \leftarrow$ BUILD–TREE$(\mathcal{T}_k, \phi_{m+1}, A, t)$
12. update the schedule $\mathcal{S} \leftarrow \mathcal{S} \cup A$ and lifetime $T \leftarrow T + 1$

Fig. 6. A high level description of the GREEDY CMLDA heuristic.

loop is dominated by the running time of the BUILD-TREE routine. Observe that, the worst-case running time of the BUILD-TREE routine is $O(n^3)$. Thus, the worst-case running time of Phase II is $O(n^3)$, and consequently the worst-case running time of the GREEDY CMLDA heuristic is $O(m^{15} \log m + n^3)$.

As discussed in Section 5, by appropriately choosing the number of super-sensors m, we can achieve a significant reduction in the actual running time of GREEDY CMLDA with respect to MLDA. For example, for $m = n^{3/16}$, the worst-case running time of GREEDY CMLDA is $O(n^3)$.

4.2. The incremental CMLDA heuristic

In this section, we describe an improved heuristic for the MLDA problem in large-scale sensor networks. Recall that, in solving the MLDA problem, we are essentially interested in *provisioning* the (edge) capacities of an admissible flow network G—such that the $s - t$ flow from each sensor s to the base station t in G is maximized. The proposed heuristic builds such a flow network by incrementally provisioning capacities on its edges.

Consider the set of n sensor nodes and a base station node t. Let the initial energy of each sensor be $\mathcal{E}$. Let the sensors be partitioned into m super-sensors $\phi_1, \ldots, \phi_m$, each having at most c sensors for some constant c. Let super-sensor ϕ_{m+1} consist only of the base station t. The linear relaxation of the integer program in Table 1 can be used to find an optimal admissible flow network G for the super-sensors. Our basic approach is to use flow network G in order to construct an admissible flow network for the sensors. The INCREMENTAL MLDA heuristic consists of four phases and works as follows.

Phase I. In the first phase, we assign the initial energy of each super-sensor ϕ_i $(i = 1, 2, \ldots, m)$ to be the sum of the initial energies of the sensors within it, i.e. $\mathcal{E}_{\phi_i} = \mathcal{E}|\phi_i|$. The distance between two super-sensors ϕ_i and ϕ_j is assigned to be the maximum distance between any two nodes (sensor or base station) u and v, such that $u \in \phi_i$ and $v \in \phi_j$. Having set up the initial energies and the distances between the super-sensors, we compute an optimal admissible flow network G of maximum lifetime T for the super-sensors $\phi_1, \ldots, \phi_m$ (with the base station ϕ_{m+1}), [3] using the linear relaxation of the integer program in Table 1. Specifically, we provision the capacity f_{ϕ_i,ϕ_j} between

[3] For brevity, we refer to both t and ϕ_{m+1} as the base station.

every pair of super-sensors ϕ_i and ϕ_j, such that the system of super-sensors has a lifetime T.

Phase II. Given the flow network G for the super-sensors and the target lifetime T computed in Phase I, we next determine the capacity provisions that are required for each of the sensors within every super-sensor.

Consider a super-sensor ϕ_i. In order to live for T rounds, each sensor s in ϕ_i needs to transmit T packets to the base station t (ϕ_{m+1}). Observe that, each super-sensor ϕ_i has a maximum flow of size $\geqslant T$ to ϕ_{m+1} in G. Stated otherwise, each super-sensor ϕ_i has a minimum $\phi_i - \phi_{m+1}$ cut of size $\geqslant T$ in G. Therefore, for each sensor s in ϕ_i, we need to determine the capacity provisions between s and the remaining sensors in ϕ_i, as well as between s and each of the super-sensors $\phi_1, \ldots, \phi_{i-1}$, $\phi_{i+1}, \ldots, \phi_{m+1}$, such that

(i) the sum of the provisioned capacities from (to) all the sensors in ϕ_i to (from) each super-sensor ϕ_j equals the provisioning f_{ϕ_i,ϕ_j} (f_{ϕ_j,ϕ_i}) obtained from Phase I, and
(ii) each sensor s in ϕ_i can push T packets (flow) to the remaining super-sensors. Note that this guarantees that each $s - \phi_{m+1}(s-t)$ minimum cut in the resulting flow network has capacity $\geqslant T$, since each minimum $\phi_i - \phi_{m+1}$ cut in G has capacity $\geqslant T$.

In doing so, our objective is to *minimize* the maximum energy consumed by any sensor within the super-sensor ϕ_i, thereby extending the lifetime of the sensors. The necessary provisions can be computed in polynomial-time using the linear program in Table 2.

For each super-sensor $\phi_1, \ldots, \phi_m$, the linear program minimizes the maximum energy consumed by any sensor within the super-sensor, subject to the energy constraint (16), the capacity constraints (17) and (18), and the flow constraints (19)–(23). For each sensor $s \in \phi_i$, constraint (16)

Table 2
Linear program for Phase II of INCREMENTAL CMLDA

Objective:

$$\text{minimize } \mathscr{E}_{\max} \tag{15}$$

Energy and capacity constraints:

$$\sum_{j\in\phi_i} f_{s,j}TX_{s,j} + \sum_{j\in\phi_i} f_{j,s}RX_s + \sum_{j\neq i} f_{s,\phi_j}TX_{s,j} + \sum_{j\neq i} f_{\phi_j,s}RX_s \leqslant \mathscr{E}_{\max}, \quad \forall s \in \phi_i, \tag{16}$$

$$\sum_{s\in\phi_i} f_{s,\phi_j} = f_{\phi_i,\phi_j}, \quad \forall j \neq i, \tag{17}$$

$$\sum_{s\in\phi_i} f_{\phi_j,s} = f_{\phi_j,\phi_i}, \quad \forall j \neq i. \tag{18}$$

Flow constraints:

$$\sum_{j\in\phi_i} \pi^{(s)}_{j,k} = \sum_{j\in\phi_i} \pi^{(s)}_{k,j} + \sum_{j\neq i} \pi^{(s)}_{k,\phi_j}, \quad \forall k \in \phi_i \quad \text{and} \quad k \neq s, \tag{19}$$

$$\sum_{j\in\phi_i} \pi^{(s)}_{j,s} + T = \sum_{j\in\phi_i} \pi^{(s)}_{s,j} + \sum_{j\neq i} \pi^{(s)}_{s,\phi_j}, \tag{20}$$

$$0 \leqslant \pi^{(s)}_{k,j} \leqslant f_{k,j}, \quad \forall j \in \phi_i \quad \text{and} \quad \forall k \in \phi_i, \tag{21}$$

$$0 \leqslant \pi^{(s)}_{k,\phi_j} \leqslant f_{k,\phi_j}, \quad \forall j \neq i, \tag{22}$$

$$\sum_{j\neq i, k\in\phi_i} \pi^{(s)}_{k,\phi_j} = T, \tag{23}$$

where $s \in \phi_i$ and all variables are required to be non-negative.

gives the energy consumed at the sensor. Constraints (17) and (18) ensure that the sum of the provisioned capacities from (to) all the sensors in ϕ_i to (from) each super-sensor ϕ_j equals the provisioning f_{ϕ_i,ϕ_j} (f_{ϕ_j,ϕ_i}). Constraints (19) and (20) enforce the flow conservation principle at a sensor; constraints (21) and (22) ensure that the capacity constraints on the edges of the flow network are respected; and constraint (23) ensures that each sensor $s \in \phi_i$ has a minimum cut of size T to the remaining super-sensors.

Phase III. From Phase II, we obtain the capacity provisions between any sensor s and all other sensors in the same super-sensor. In addition, we know the provisions that are required between s and each of the remaining super-sensors. However, in order to construct the admissible flow network for the sensors, we need to determine the capacities that need to be provisioned between individual sensors in different super-sensors.

In particular, consider any two distinct super-sensors ϕ_l and ϕ_r $(l, r = 1, 2, \ldots, m+1)$. [4] For each sensor i in ϕ_l, we know the capacity f_{i,ϕ_r} ($f_{\phi_r,i}$) provisioned from (to) sensor i to (from) super-sensor ϕ_r. Similarly, for each sensor j in ϕ_r, we know the capacity f_{j,ϕ_l} ($f_{\phi_l,j}$) provisioned from (to) sensor j to (from) ϕ_l. To complete the construction, we provision capacities between pairs of sensors from ϕ_l and ϕ_r, while ensuring that

(i) the total capacity provisioned from (to) each sensor $i \in \phi_l$ to (from) all the sensors in ϕ_r equals the provisioning f_{i,ϕ_r} ($f_{\phi_r,i}$) obtained from Phase II, and
(ii) the total capacity provisioned from (to) each sensor $j \in \phi_r$ to (from) all the sensors in ϕ_l equals the provisioning f_{j,ϕ_l} ($f_{\phi_l,j}$).

In doing so, we again maintain the objective of minimizing the maximum energy consumed by each sensor in any of the super-sensors. This is a *matching* problem which can be solved in polynomial-time using the linear program in Table 3.

[4] We do not provision capacities from the base station t to any sensor.

For each pair of super-sensors ϕ_l and ϕ_r, the linear program minimizes the maximum energy consumed by any sensor within the super-sensors, subject to the constraints (25)–(30). For each sensor $i \in \phi_l$ and each sensor $j \in \phi_r$, constraints (25) and (28) give the energy consumed at the respective sensors. Constraints (26) and (27) guarantee that the total capacity provisioned from (to) each sensor $i \in \phi_l$ to (from) all the sensors in ϕ_r equals the provisioning f_{i,ϕ_r} ($f_{\phi_r,i}$). Similarly, constraints (29) and (30) ensure that the total capacity provisioned from (to) each sensor $j \in \phi_r$ to (from) all the sensors in ϕ_l equals the provisioning f_{j,ϕ_l} ($f_{\phi_l,j}$).

Phase IV. At the end of Phase III, we obtain an admissible flow network with lifetime T for all n sensors and the base station t with capacities provisioned between them. Note that, these capacities are fractional non-negative numbers. In order to obtain a data gathering schedule for the sensors, we first compute the the maximum energy $\mathscr{E}_{\max}$ consumed by any sensor. Then, using Lemma 4 below, we scale the provisioned capacities by a factor of $\alpha = \mathscr{E}/\mathscr{E}_{\max}$, and thus obtain an admissible flow network G_0 with lifetime $T_0 = \alpha T$ that respects the initial energy of the sensors.

Lemma 4. *Given the location of n sensors and a base station t, the lifetime of an admissible flow network scales linearly with the energy of the sensors.*

Proof. Follows directly from the energy constraint (5) for the sensors in the formulation of the MLDA problem in Table 1. □

Next, we floor all the capacities of G_0 to obtain an admissible flow network with integer capacities (flows). We call this flow network the INCREMENTAL CMLDA flow network. Using this flow network, we finally compute the integral system lifetime (as in the MLDA algorithm), and a data gathering schedule $\mathscr{S}$ using the GETSCHEDULE algorithm from Section 3.

Fig. 7 provides an illustration of the INCREMENTAL CMLDA heuristic. We consider a set of 4 sensors and a base station labeled 5, clustered into super-sensors Q_1, Q_2 and Q_3. Fig. 7(a) shows an

Table 3
Linear program for Phase III of INCREMENTAL CMLDA

Objective:

$$\text{minimize } \mathscr{E}_{\max} \tag{24}$$

Energy and capacity constraints for ϕ_l:

$$\sum_{j\in\phi_r} f_{i,j}TX_{i,j} + \sum_{j\in\phi_r} f_{j,i}RX_i \leqslant \mathscr{E}_{\max}, \quad \forall i \in \phi_l, \tag{25}$$

$$\sum_{j\in\phi_r} f_{i,j} = f_{i,\phi_r}, \quad \forall i \in \phi_l, \tag{26}$$

$$\sum_{j\in\phi_r} f_{j,i} = f_{\phi_r,i}, \quad \forall i \in \phi_l. \tag{27}$$

Energy and capacity constraints for ϕ_r:

$$\sum_{j\in\phi_l} f_{i,j}TX_{i,j} + \sum_{j\in\phi_l} f_{j,i}RX_i \leqslant \mathscr{E}_{\max}, \quad \forall i \in \phi_r, \tag{28}$$

$$\sum_{j\in\phi_l} f_{i,j} = f_{i,\phi_l}, \quad \forall i \in \phi_r, \tag{29}$$

$$\sum_{j\in\phi_l} f_{j,i} = f_{\phi_l,i}, \quad \forall i \in \phi_r, \tag{30}$$

where all variables are required to be non-negative integers.

optimal admissible flow network for the super-sensors with lifetime 100. We use this flow network to provision capacities for the sensors within each super-sensor, during Phase II of the heuristic. The resulting network is shown in Fig. 7(b). Observe that, sensors 1 and 2 in Q_1, each have a min-cut of size 100 to super-sensors Q_2 and Q_3. Similarly, sensors 3 and 4 in Q_2, each have a min-cut of size 100 to super-sensors Q_1 and Q_3. Finally, we solve the matching problem in Phase III of the heuristics, to determine the capacity provisioning between pairs of sensors from Q_1 and Q_2. The INCREMENTAL CMLDA flow network for the sensors with lifetime of 100 rounds is shown in Fig. 7(c). Note that each sensor has a min-cut of size 100 to the base station 5.

Worst-case running time of INCREMENTAL CMLDA. The INCREMENTAL CMLDA heuristic consists of four phases. Phase I involves solving the linear program (4), which has $O(m^3)$ variables, and thus the worst-case running time of Phase I is $O(m^{15}\log m)$ (see Lemma 1 and Theorem 3). In Phase II, we solve m instances of the linear program (4), each with $O(cm^2)$ variables. Each instance of the linear program (4) can be solved in worst-case running time $O(c^5m^{10}\log(cm))$, using Lemma 1. Hence, the worst-case running time of Phase II is $O(c^5m^{11}\log(cm))$. In Phase III, we solve the linear program (4) with $O(c^2)$ variables in worst-case running time $O(c^{10}\log c)$. Finally, Phase IV has worst-case running time $O(n^2)$, since we need to find the maximum energy consumed by any sensor.

Therefore, the total worst-case running time of INCREMENTAL CMLDA is

$$O(m^{15}\log m) + O(c^5m^{11}\log(cm)) + O(c^{10}\log c) + O(n^2). \tag{31}$$

As discussed in Section 5, by appropriately choosing the number of super-sensors m and the maximum number c of sensors within any super-sensor, we can achieve a significant reduction in the actual running time of INCREMENTAL CMLDA with respect to the MLDA. For example, for $m = n^{5/11}$, the worst-case running time of the INCREMENTAL CMLDA is $O(n^{75/11}\log n) = O(n^{6.8}\log n)$.

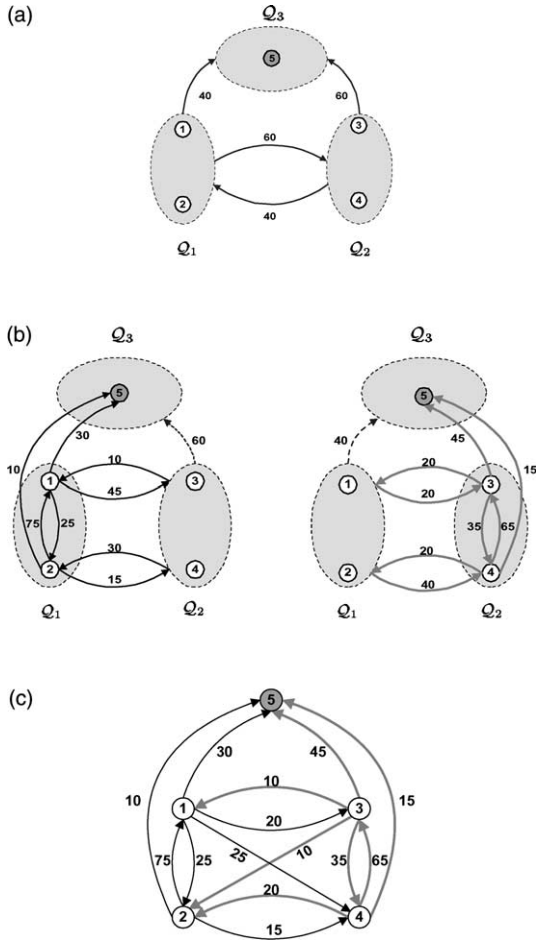


Fig. 7. Illustrative example of the INCREMENTAL CMLDA heuristic: (a) an admissible flow network from Phase I for super-sensors Q_1 and Q_2 with base station in Q_3, (b) capacity provisioning from Phase II for sensors 1, 2 in super-sensor Q_1 and sensors 3, 4 in super-sensor Q_2 and (c) the INCREMENTAL CMLDA flow network for the sensors with lifetime of 100 rounds.

5. Experiments

In this section, we compare the data gathering schedule given by the near-optimal MLDA algorithm, the GREEDY CMLDA heuristic and the INCREMENTAL CMLDA heuristic, with that obtained from a chain-based 3-level hierarchical protocol proposed by Lindsey, Raghavendra and Sivalingam [15]. For brevity, we refer to this protocol as the LRS protocol. We choose this protocol since it outperforms other competitive protocols (e.g. LEACH [8]) in terms of system lifetime.

LRS protocol for constructing a data gathering schedule. In this protocol, sensor nodes are initially grouped into clusters based on their distances from the base station. A chain is formed among the sensor nodes in a cluster at the lowest level of the hierarchy. Gathered data, moves from node to node, gets aggregated, and reaches a designated leader in the chain i.e. the cluster head. At the next level of the hierarchy, the leaders from the previous level are clustered into one or more chains, and the data is collected and aggregated in each chain in a similar manner. Thus, for gathering data in each round, each sensor transmits to a close neighbor in a given level of the hierarchy. This occurs at every level, the only difference being that nodes that are receiving at each level are the only nodes that rise to the next level in the hierarchy. Finally at the top level, there is a single leader node transmitting to the base station. To increase the lifetime of the system, the leader in each chain is chosen in a round-robin manner in each round. Observe that, the manner in which chain leaders are selected in each level of the hierarchy, naturally defines an aggregation tree, for each round of data gathering.

For the initial set of experimental results, we consider a network of sensors randomly distributed in a 50 m $\times$ 50 m field. The number of sensors in the network, i.e. the network size n, is varied to be 40, 50, 60, 80 and 100 respectively. Each sensor has an initial energy of 1 J and the base station is located at (25 and 150 m). Each sensor generates packets of size 1000 bits. The energy model for the sensors is based on the first order radio model described in Section 2.

Each experiment corresponds to a random placement of the sensors, for a particular network size. In each experiment, we measure the lifetime T, i.e. the number of rounds before the first sensor is drained of its energy, for the data gathering schedule given by the LRS protocol. For the same placement of sensors, we measure the lifetime of the data gathering schedules obtained from MLDA, GREEDY CMLDA (G-CMLDA) and INCREMENTAL CMLDA (I-CMLDA). We define the *performance ratio* R_M as the ratio of the system lifetime

achieved using MLDA to the lifetime given by the LRS protocol. Similarly, we define $R_G(R_I)$ to be the ratio of the system lifetime achieved using the GREEDY CMLDA (INCREMENTAL MLDA) heuristic to the lifetime given by the LRS protocol. Recall that, the (integral) solution given by MLDA is an approximation of the optimal fractional solution. We denote OPT to be the optimal system lifetime for any particular experiment.

For a data gathering schedule $\mathscr{S}$, we define the *depth* of a sensor v to be its average number of hops from the base station in the schedule, i.e. the average of its depths in each of the aggregation trees in $\mathscr{S}$. The depth of the schedule is defined as $\max\{\text{depth}(i)\}$, among all sensors i in the network. We measure the depth D of a schedule constructed using each of the MLDA, GREEDY (INCREMENTAL) CMLDA and LRS algorithms. Note that, the depth of a data gathering schedule is an interesting metric since it gives an estimate of the (maximum) average delay [5] that is incurred in sending data packets from any sensor to the base station.

Finally, for the GREEDY (INCREMENTAL) CMLDA heuristics, we denote c to be the number of sensors in a cluster (super-sensor). Given the location of the sensors and the base station, we employ a greedy clustering algorithm similar to the the chain-forming algorithm used by the LRS protocol [15]—pick a sensor i farthest from the base station and form a cluster that includes i and its $c-1$ nearest neighbors; continue the process with the remaining sensors until all sensors have been included in some cluster. For a particular network size, we assign the size of a cluster in GREEDY (INCREMENTAL) CMLDA to be identical to the size of a chain in the LRS protocol. By clustering the sensors in the above manner, we can efficiently compute maximum lifetime schedules for the super-sensors in large networks. Further, with a proper choice of the number of super-sensors and the size of each super-sensor, we can solve the linear programs in Phases II and III of the INCREMENTAL CMLDA heuristic in a fast and efficient manner. Observe that, the MLDA algorithm and the GREEDY (INCREMENTAL) CMLDA heuristics presented in this paper are essentially centralized in nature. This implies that the clustering of the sensors need to be pre-computed at the base station. Similarly, an appropriate data gathering schedule is pre-computed at the base station (which is less likely to be resource-constrained) and transmitted to the individual sensors. We take advantage of the fact that the base-station is aware of the locations of the sensors and have sufficient processing capabilities to compute efficient data-gathering schedule(s) for the sensors.

Tables 4 and 5 summarize our main results. Note that the presented values for lifetime and depth are averaged across 20 different experiments for each network size. Further, the MIN and MAX columns for R_M, R_G and R_I indicate the corresponding minimum and maximum performance ratios observed from those experiments. We make the following key observations for the smaller sensor networks:

- The lifetime of a schedule obtained using the INCREMENTAL CMLDA heuristic is always within 3% of the optimal fractional solution, while the lifetime of a schedule given by the GREEDY CMLDA heuristic is always within 9% of the optimal solution.
- The lifetime of a schedule given by the MLDA algorithm is near-optimal. In fact, the approximation scheme in Section 3 leads to a reduction in the (integral) system lifetime by no more than three rounds, when compared to the optimal solution.
- The MLDA algorithm significantly outperforms the LRS protocol in terms of system lifetime. Further, the GREEDY CMLDA heuristic performs 1.10–2.16 times better than LRS, while the INCREMENTAL CMLDA heuristic performs 1.15–2.24 times better than LRS.
- The average depth of a data gathering schedule attained by the GREEDY (INCREMENTAL) CMLDA heuristic is slightly higher than that of the LRS protocol. Note that the three level protocol in LRS is specifically devised to reduce the average depth of each sensor [15]. To this end, the GREEDY (INCREMENTAL) CMLDA heuristic does quite well in attaining comparable

[5] On a 1 Mbps link, a 1000 bit message can incur a delay of 1 ms on each hop to the base station.

Table 4
Experimental results for 50 m × 50 m sensor network

Input		MLDA			G-CMLDA		I-CMLDA		LRS	
n	*c*	OPT	*T*	*D*	*T*	*D*	*T*	*D*	*T*	*D*
40	5	6611.8	6610	4.9	6442	4.6	6512	4.8	5592	4.4
50	5	6809.0	6808	5.8	6747	5.6	6786	5.5	5466	5.1
60	5	7176.2	7174	6.2	6896	6.0	7084	6.3	5872	5.2
80	10	7946.9	7945	7.5	7509	6.6	7809	6.9	6002	6.1
100	10	8292.6	8290	8.2	8011	7.2	8121	7.8	5526	6.6

Table 5
Performance ratios for 50 m × 50 m sensor network

Input		R_M		R_G		R_I	
n	*c*	MIN	MAX	MIN	MAX	MIN	MAX
40	5	1.15	1.48	1.10	1.42	1.15	1.45
50	5	1.20	1.90	1.20	1.65	1.20	1.72
60	5	1.18	1.66	1.18	1.62	1.16	1.64
80	10	1.27	2.05	1.21	2.01	1.27	1.96
100	10	1.42	2.32	1.33	2.16	1.38	2.24

sensor depths, while delivering significant improvements in system lifetime.

For each of the above experiments, we also measured its total running time (in terms of CPU time) when using MLDA, GREEDY CMLDA and INCREMENTAL CMLDA respectively. Table 6 shows the average CPU times for different network sizes. Observe that, each of these experiments were conducted using MATLAB implementations of the algorithms on typical machine configurations of 866 MHz (Intel) processor/1GB physical memory. As shown in Table 6, the INCREMENTAL CMLDA heuristic has an average running time that is approximately 33% of that for MLDA, while the GREEDY CMLDA heuristic has an average running time that is about 10% of that for MLDA.

For our next set of experiments, we consider larger networks of sensors randomly distributed in a 100 m × 100 m field. The number of sensors in the network, i.e. the network size *n*, is varied to be 100, 200, 300 and 400 respectively. Each sensor has an initial energy of 1 J and the base station is located at (50 and 300 m). Once again, the presented values for lifetime and depth are averaged across 20 different experiments for each network size. Due to the high complexity of the algorithm, we do not include any results regarding the performance of MLDA for the large-scale networks. The clustering in CMLDA (chain formation in LRS) is done in the manner described above. We summarize our results in Tables 7 and 8.

Table 6
Comparison of CPU times (in min) for MLDA, GREEDY CMLDA and INCREMENTAL CMLDA

Input	MLDA	G-CMLDA	I-CMLDA
40	6.52	11.8	12.5
50	48.0	16.8	22.9
60	108.5	17.2	48.1
80	252.6	24.1	80.8
100	335.6	29.2	106.5

We make the following observations:

- The GREEDY CMLDA and INCREMENTAL CMLDA heuristics significantly outperform the LRS protocol in terms of system lifetime. In particular, the GREEDY CMLDA heuristic delivers system lifetimes that are 1.20–2.27 times larger than LRS; while the INCREMENTAL CMLDA heuristic attains lifetimes that are 1.22–2.61 times larger than LRS.
- The INCREMENTAL CMLDA heuristic obtains systems lifetimes that are 10–20% larger than

Table 7
Experimental results for 100 m × 100 m sensor network

INPUT		G-CMLDA		I-CMLDA		LRS	
n	*c*	*T*	*D*	*T*	*D*	*T*	*D*
100	10	3200	7.5	3611	7.8	2458	6.9
200	10	4086	10.6	4512	10.6	2854	9.6
300	15	4858	13.2	5560	13.1	3212	12.1
400	20	5202	20.6	6142	19.8	3654	18.6
500	25	5533	26.1	6577	25.8	3596	24.8

Table 8
Performance ratios for 100 m × 100 m sensor network

INPUT		R_G		R_I	
n	*c*	MIN	MAX	MIN	MAX
100	10	1.32	1.91	1.22	2.61
200	10	1.20	2.27	1.30	2.43
300	15	1.29	2.13	1.50	2.41
400	20	1.31	1.88	1.40	2.02
500	25	1.33	2.09	1.36	2.33

the GREEDY CMLDA heuristic. Observe that, this improvement comes at the cost of increased complexity involved with INCREMENTAL CMLDA.

- The average depth of a data gathering schedule attained by the GREEDY (INCREMENTAL) CMLDA heuristic is only slightly higher than that of the LRS protocol.

In conclusion, our experimental results demonstrate that the clustering-based heuristics can achieve as much as a factor of 2.61 increase in the lifetime of large-scale sensor networks, when compared to the LRS protocol, and incurs a small increase in the delay experienced by individual sensors. Thus, while it is true that our algorithms are centralized, the payback is significant in terms of improvement in the system lifetime.

6. Conclusions

In this paper, we proposed a polynomial-time near-optimal algorithm (MLDA) for solving the maximum lifetime data gathering problem for sensor networks, when the sensors are allowed to perform in-network aggregation of data packets. Given the complexity of the MLDA algorithm, we next described efficient clustering-based heuristics to solve the maximum lifetime data aggregation problem in large sensor networks. Further, we presented experimental results demonstrating that the proposed methods attain significant improvements in system lifetime, when compared to existing protocols.

There are a number of important issues related to the maximum lifetime data gathering problem that need to be investigated in the future. In the work presented in this paper, we make the simplistic assumption that a sensor can always aggregate its own data packets with those of any other sensor in the network. As part of our current research, we are exploring a more complex scenario where a sensor is permitted to aggregate its own packets with only certain sensors, while acting as a router for other incoming packets. In the future, we plan to investigate modifications to the MLDA algorithm that would allow sensors to be added to (removed from) the network, without having to re-compute the entire schedule. Further, we plan to study the data gathering problem with depth (delay) constraints for individual sensors, in order to attain desired tradeoffs between the delay experienced by the sensors and the lifetime achieved by the system.

Appendix A. Proof of Theorem 2

The proof is based on a powerful theorem in graph theory [6,16].

Preliminaries. A *multi-graph* is a graph which may have parallel edges, i.e. it may have multiple edges with the same end-vertices. An *arborescence* $\mathscr{A}$ rooted at a vertex t is a a directed graph such that there is a unique directed path from t to each vertex in $\mathscr{A}$. A *spanning arborescence* of a multi-graph is an arborescence that spans all its vertices. The *transpose* G^{T} of a directed graph G is the graph that results from G by reversing the directions of all of its edges. Given a graph $G=(V,E)$ and an edge $e \in V \times V$, let $G+e$ denote the graph resulting from G by adding e to it.

Consider a directed graph $G=(V,E)$ with positive integer capacities $c(e)$ assigned to its edges $e \in E$. Let $\widehat{G}$ be the graph that results from G by replacing each one of its edges $e=(u,v)$ with $c(e)$ parallel edges (u,v); for the multi-graph $c(e)=c(u,v)$ will denote the number of parallel edges (u,v).

For any subset $S \subset V$, let $(S,\overline{S})$, called a *cut*, denote the set of edges in G from S to $\overline{S}=V-S$. The *capacity* (*size*) of a cut $(S,\overline{S})$, denoted with $\omega_G(S)$, is sum of the capacities of the edges in $(\overline{S},S)$; the capacity $\omega_{\widehat{G}}(S)$ of the cut $(S,\overline{S})$ with respect to $\widehat{G}$ equals the total number of edges in $(S,\overline{S})$ in $\widehat{G}$. A $u-v$ cut is a cut $(S,\overline{S})$ such that $u \in S$ and $v \in \overline{S}$. A $u-v$ min-cut is a $u-v$ cut with minimum capacity.

For any subgraph H of G let $G-H$ denote the graph resulting from G by deleting the edges in H. Similarly, let $\widehat{G}-\lambda H$ denote the graph resulting from the multi-graph $\widehat{G}$ by deleting λ copies of H from it, $0 \leqslant \lambda \leqslant \min\{c(e): e \in H\}$. Note that the operation $\widehat{G}-\lambda H$ corresponds to a reduction of the capacities of the edges of G, which are also in H, by λ; thus, for convenience, we also denote $\widehat{G}-\lambda H$ by $G-\lambda H$.

Definition 3. Consider a multi-graph $\widehat{G}=(V,\widehat{E})$. Let t be a vertex in V, and k a positive integer. Suppose that, for all $v \in V$, each $t-v$ cut has size $\geqslant k$. An arborescence $\mathscr{A}=(\mathscr{V}_{\mathscr{A}},\mathscr{E}_{\mathscr{A}})$ of the multi-graph $\widehat{G}$ is called *admissible* if for every $S \subset V$, with $t \in S$, $\omega_{\widehat{G}-\mathscr{A}}S \geqslant k-1$.

Note that an arborescence with no edges is admissible.

Lemma A.1. *Consider a graph $G=(V,E)$ with capacities associated with its edges. Let $(S,\overline{S})$ be any cut in G. Then the capacity of $(S,\overline{S})$ in G is equal to the capacity of $(\overline{S},S)$ in G^{T} and vice-versa.*

Proof. Follows directly from the definition of cut capacities, and the definition of the transpose of a graph. □

Theorem A.1 (Edmonds [6] and Lovász [16]). *Let $\widehat{G}=(V,\widehat{E})$ be a multi-graph, t be a vertex in V, and k be a positive integer. If, for all $v \in V$, each $t-v$ cut has size $\geqslant k$, then $\widehat{G}$ has k edge-disjoint spanning arborescences rooted at t.*

Moreover, the following corollary follows from the construction in the proof of Theorem A.1.

Corollary A.1. *Let $\widehat{G}=(V,\widehat{E})$ be a multi-graph, t be a vertex in V, and k a positive integer. Suppose that, for all $v \in V$, each $t-v$ cut has size $\geqslant k$. Let $\mathscr{A}=(\mathscr{V}_{\mathscr{A}},\mathscr{E}_{\mathscr{A}})$ be an admissible arborescence of $\widehat{G}$. If $\mathscr{A}$ is not spanning V, there exists an edge $e \in (V_{\mathscr{A}},\overline{V_{\mathscr{A}}})$ such that $\mathscr{A}$ together with e is also an admissible arborescence.*

Proof sketch. Follows directly from the construction in the proof of Theorem A.1. □

Proof of Theorem 2. Since G is admissible, each sensor node $v \in V$ has a $v-t$ cut of capacity $\geqslant T$ to the base station t.

Observe that, the transpose $\mathscr{A}^{\mathrm{T}}$ of a spanning admissible arborescence $\mathscr{A}$ of the multi-graph that corresponds to G^{T}, provides us with an aggregation tree for G for aggregating 1 value from each sensor. Moreover, from Corollary A.1 we know that a spanning admissible arborescence for the multi-graph $\widehat{G}^{\mathrm{T}}$ can be constructed using a simple greedy algorithm. Specifically, start with an arborescence $\mathscr{A}$ that contains just t, and while $\mathscr{A}$ is not spanning V, add an edge $e \in (V_{\mathscr{A}},\overline{V_{\mathscr{A}}})$ such that $\mathscr{A}+e$ is admissible for $\widehat{G}^{\mathrm{T}}$. Thereby, it is easy to see that the GetTree algorithm can always find

an aggregation tree A for collecting one data packet from each sensor, such that the $(A,1)$-reduction of G is feasible. Further, if the (A,f)-reduction of G is feasible for some $f1$, we can use the aggregation tree A to collect f data packets from each sensor. This results in fewer aggregation trees and hence an improvement in the running time of our algorithm. Finally, we can iteratively invoke the GetTree procedure to find a sequence of aggregation trees for collecting T data packets from each of the sensors. □

References

[1] D. Bertsimas, J.N. Tsitsiklis, Introduction to Linear Optimization, Athena Scientific, Belmont, MA, 1997.

[2] M. Bhardwaj, T. Garnett, A.P. Chandrakasan, Upper bounds on the lifetime of sensor networks, in: Proceedings of International Conference on Communications, 2001.

[3] J.H. Chang, L. Tassiulas, Energy conserving routing in wireless ad-hoc networks, in: Proceedings of IEEE INFOCOM, 2000.

[4] J.H. Chang, L. Tassiulas, Maximum lifetime routing in wireless sensor networks, in: Proceedings of Advanced Telecommunications and Information Distribution Research Program, College Park, MD, 2000.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Max-flow min-cut theorem, in: Introduction to Algorithms, MIT Press, Cambridge, MA, 1998.

[6] J. Edmonds, Edge-disjoint branchings, in: Combinatorial Algorithms, Academic Press, London, 1973.

[7] A.V. Goldberg, S. Rao, Beyond the flow decomposition barrier, in: Proceedings of 38th IEEE Annual Symposium on Foundation of Computer Science, 1997.

[8] W. Heinzelman, A.P. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocols for wireless microsensor networks, in: Proceedings of Hawaiian International Conference on Systems Science, 2000.

[9] W. Heinzelman, J. Kulik, H. Balakrishnan, Adaptive protocols for information dissemination in wireless sensor networks, in: Proceedings of 5th ACM/IEEE Mobicom Conference, 1999.

[10] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, in: Proceedings of 6th ACM/IEEE Mobicom Conference, 2000.

[11] J.M. Kahn, R.H. Katz, K.S.J. Pister, Mobile networking for smart dust, in: Proceedings of 5th ACM/IEEE Mobicom Conference, 1999.

[12] B. Krishnamachari, D. Estrin, S. Wicker, The impact of data aggregation in wireless sensor networks, in: Proceedings of International Workshop on Distributed Event-Based Systems, 2002.

[13] X. Lin, I. Stojmenovic, Power-aware routing in ad hoc wireless networks, University of Ottawa, TR-98-11, 1998.

[14] S. Lindsey, C.S. Raghavendra, PEGASIS: power efficient gathering in sensor information systems, in: Proceedings of IEEE Aerospace Conference, 2002.

[15] S. Lindsey, C.S. Raghavendra, K. Sivalingam, Data gathering in sensor networks using the energy*delay metric, in: Proceedings of the IPDPS Workshop on Issues in Wireless Networks and Mobile Computing, 2001.

[16] L. Lovász, On two minimax theorems in graph theory, Journal of Combinatorial Theory Series B 21 (1976) 96–103.

[17] S. Madden, R. Szewczyk, M.J. Franklin, D. Culler, Supporting aggregate queries over ad-hoc wireless sensor networks, in: Proceedings of 4th IEEE Workshop on Mobile Computing and Systems Applications, 2002.

[18] R. Min, M. Bhardwaj, S.H. Cho, A. Sinha, E. Shih, A. Wang, A.P. Chandrakasan, Low-power wireless sensor networks, in: VLSI Design, 2001.

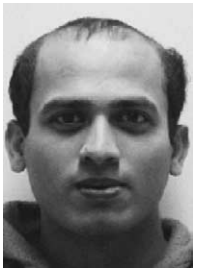
[19] J. Rabaey, J. Ammer, J.L. da Silva Jr., D. Patel, PicoRadio: ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes, in: Proceedings of the IEEE Computer Society Annual Workshop on VLSI, 2000.

[20] S. Singh, M. Woo, C. Raghavendra, Power-aware routing in mobile ad hoc networks, in: Proceedings of 4th ACM/IEEE Mobicom Conference, 1998.

Konstantinos Kalpakis received the B.S. degree in Computer Engineering and Informatics from the University of Patras, Greece, in 1989, and the M.S. and Ph.D. degrees in Computer Science from the University of Maryland Graduate School, Baltimore, in 1992 and 1994 respectively. He is an Associate Professor in the Computer Science and Electrical Engineering Department at the University of Maryland Baltimore County since the Fall of 1996. He was a visiting computer scientist at the National Institute of Standards and Technology and a research assistant professor at UMBC from 1994 until 1996, working in the areas of digital libraries and electronic commerce.

His research interests are in the areas of parallel and distributed systems, digital libraries and electronic commerce, database systems, and combinatorial optimization. He is a member of the ACM, IEEE Computer Society, Upsilon Pi Epsilon, and an associate member of Sigma Xi.

Koustuv Dasgupta received his B.S. degree in Computer Science and Engineering from Jadavpur University, India in 1997. He received his M.S. in Computer Science from the University of Maryland, Baltimore County in 2000 and is currently pursuing his Ph.D. at the same University. His research interests include distributed systems, data replication and Internet technologies.



Parag Namjoshi received his B.E. in Computer Engineering from Pune University in 1998, and the M.S. in Computer Sciences from the University of Maryland, Baltimore County in 2002. Currently, he is pursuing the Ph.D. degree in computer sciences at the University of Maryland, Baltimore County. His research interests include wireless sensor networks, mobile computing and distributed systems.