Energy-Efficient Data Redistribution in Sensor Networks

Bin Tang, Neeraj Jaggi, Haijie Wu, and Rohini Kurkal
Department of Electrical Engineering and Computer Science
Wichita State University, Wichita, KS 67260
bintang@cs.wichita.edu, neeraj.jaggi@wichita.edu, hxwu1@wichita.edu, rxkurkal@wichita.edu

***Abstract*—We address the *energy-efficient data redistribution problem* in data intensive sensor networks (DISNs). The key question in sensor networks with large volumes of sensory data is how to redistribute the data efficiently under limited storage and energy constraints at the sensor nodes. The goal of the redistribution scheme is to minimize the energy consumption during the process, while guaranteeing full utilization of the distributed storage capacity in the DISNs. We formulate this problem as a minimum cost flow problem, which can be solved optimally. However, the optimal solution's centralized nature makes it unsuitable for large-scale distributed sensor networks. We thus design a distributed algorithm for the data redistribution problem which performs very close to the optimal, and compare its performance with various intuitive heuristics. Our proposed algorithm relies on potential function based computations, incurs limited message and computational overhead at both the sensor nodes and data generator nodes, and is easily implementable in a distributed manner. We analytically show the convergence of our algorithm, and demonstrate its near-optimal performance and scalability under various network scenarios considered. Finally, we implement our distributed algorithm in TinyOS and evaluate it using TOSSIM simulator, and show that it outperforms EnviroStore, the only existing scheme for data redistribution in sensor networks, in both solution quality and overhead messages.**

Keywords – Data Redistribution, Sensor Networks

I. Background and Motivation

It has become a reality that the sensor network applications are no longer limited to just ambient sensing (e.g., light or temperature) or environmental and weather monitoring. With the emergence of a rich collection of sensory sources such as video cameras, microphones, RFID readers, telescopes and seismometers, a whole new array of data-intensive sensing applications have been researched and developed recently. They include multimedia surveillance networks [11], visual and acoustic sensor networks [17, 26], underwater or ocean seismic sensor networks [15, 27, 30] and geophysical monitoring [19, 31]. In such data intensive sensor networks (DISNs), large amount of scientific data are generated from some sensor nodes (called *data generators*[1]), stored in the network first, and collected later for further analysis.

Despite the advances in large lower-power flash memory such as parallel NAND flash technology [1], storage is still a serious resource constraint in DISNs. According to [17], an acoustic sensor that has a 1GB flash memory and is designed

[1]For the rest of the paper, we refer to non-data generator sensor nodes simply as sensor nodes.

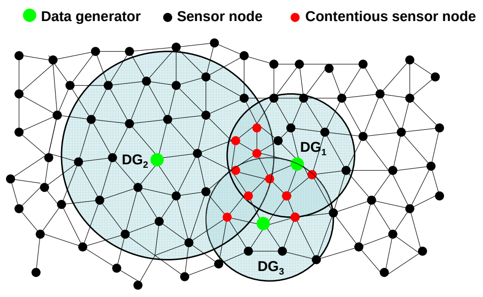

Fig. 1. Data redistribution problem with three data generators DG_1, DG_2, and DG_3. Each circle represents each data generator's offloading area (the set of sensor nodes to store the offloaded data from data generators). Redistribution contention arises when data generators have overlapping offloading areas – the sensor nodes in those areas are referred to as *contentious sensor nodes*.

to sample the entire audible spectrum will run out of its storage in just 7 hours. When the storage capacity of a sensor is reached, the data has to be redistributed/offloaded to other nodes with free storage space. Such data redistribution, if not managed well, could be a serious energy drain not only to the data generators' battery power, but to other sensor nodes involved in the redistribution process. Therefore, a major challenge in DISNs is how to store the massive amount of data inside the sensor network comprising of nodes with limited storage capacity and battery power.

In this paper, we study how to redistribute the large amount of data into the network to fully utilize the storage capacity of all the sensor nodes, while at the same time, minimizing energy consumption incurred by the data redistribution. Note that in this paper, we do not consider data retrieval (and the cost incurred), and assume that the retrieval is done using data mules [24] or by human operators manually.

Since data redistribution is energy-expensive wireless communication, it is preferred that a data generator offloads its data to other sensors closeby. When there are very few data generators distant from each other, or the amount of data to offload is small, this problem becomes trivial – each data generator can perform a breadth first search (BFS) ordering of other sensor nodes in its neighborhood with respect to distance and offload data to its one-hop neighbors first, then two-hop neighbors and so on. The sensor nodes which store the offloaded data comprise the *offloading area* of data generators and are represented as circles in Fig. 1. However, when data generators are close to each other, or the amount of generated data is comparable to the amount of available storage space in the network, *redistribution contention* arises. Fig. 1 shows three data generators DG_1, DG_2, and DG_3 with overlapping offloading areas. The challenge is how to resolve such contention while still achieving energy-efficient data redistribution.

Specifically, we formulate the data redistribution problem as a graph-theoretical problem and show that it is equivalent to the minimum cost flow problem [3, 22]. Due to the centralized nature of the optimal solution, we design a fully distributed algorithm for data redistribution which still achieves near-optimal performance. We model the sensor network as the electrostatic potential field wherein the data generators are electrical point charges. We study the data redistribution as the movement of electric particle in the potential field consequently. We also design a few centralized heuristics with lower time complexity which perform comparable to the optimal solution.

The main results and contributions of our paper include:

1) To the best of our knowledge, our work is the first one to formulate and study the data redistribution problem in sensor networks.
2) We show that the data redistribution problem is equivalent to the classic minimum cost flow problem, which can be solved optimally in polynomial time.
3) We design a fully distributed, highly scalable, and efficient data redistribution mechanism, and analytically show its convergence, near-optimal performance, and scalability under various network scenarios considered.
4) Using TOSSIM simulator, we show that our distributed algorithm significantly outperforms EnviroStore [18], the only existing data redistribution scheme in sensor networks, in terms of both solution quality and overhead messages.

Paper Organization. The rest of the paper is organized as follows. Section II discusses the related work. In Section III, we first present the network model, formalize the data redistribution problem and illustrate it with a simple example. We then show that the data redistribution problem is equivalent to the minimum cost flow problem and discuss its available centralized solutions. Section IV presents our potential field based distributed algorithm, with a few centralized heuristics. In Section V, we compare all the different algorithms, discuss the results in details, and present our insight. Section VI summarizes our results and discusses future research directions.

II. Related Work

Luo et al. [18] present a cooperative storage system for sensor networks called EnviroStore, to maximize the network's data storage capacity. They propose two data redistribution mechanisms. One is called *in-network data redistribution*, wherein data is migrated from nodes that are highly loaded in storage capacity to nodes that are not. The other is called *cross-partition data redistribution*, wherein data is offloaded from overloaded network partitions to underloaded partitions using mobile data mules. However, both data redistribution mechanisms are heuristic-based without any performance guarantees. We formulate the problem as a fundamental graph-theoretic problem and show that it can be solved optimally in a centralized way and also efficiently in a distributed manner. In this paper we only focus on the in-network data redistribution. To the best of our knowledge, EnviroStore is the only work to extensively study data redistribution in sensor networks.

A related *data migration problem* has been studied extensively in the field of parallel computing [23, 25] and disk storage [13]. It mainly studies how to schedule workload and move associated data from source processors to destination processors, or change one storage configuration into another, to better respond to the data demand changes for the purpose of load balancing. Our problem, however, is concerned with the storage space utilization as well as minimization of data redistribution energy in sensor networks.

There is literature in sensor networks that has adopted the idea of potential functions (see [29] for a good survey paper). They either study how to route packets from source to destination to avoid congestion in anycast [14] or multipath routing [20], or study the placement of mobile sinks in wireless sensor networks for energy balancing. In all those problems, there are particular traffic sources and sinks. In our data redistribution problem, we have traffic sources (data generators) while the challenge is to assign the sink nodes (offloading area). The goal is to efficiently utilize the storage capacity in order to reduce the redistribution energy cost, which is different from above problems.

Recently Gao et al. [5] developed a distributed algorithm to match critical events occurring in the sensor network to nearby available resources. The idea is to extract during preprocessing, a hierarchical well-separated tree to approximate the original network graph by a logarithmic distortion factor. Later internal nodes are used to match resources and events in its subtree. Unmatched resources or events are propagated up the tree until matched. Such preprocessing is not quite feasible in this case, and also leads to a centralized solution approach. In addition, the required redistribution scheme should be resilient towards node failures.

Our problem bears a resemblance to the graph Voronoi diagram problem [4] in the sense of "areas of influence". Graph Voronoi diagram is the graph theory equivalent of the Voronoi diagram in computational geometry. Graph Voronoi diagram characterizes regions of proximity in graphs based on shortest paths between nodes. Yet there are two differences between our data redistribution problem and graph Voronoi diagram problem. First, in data redistribution problem, the node has a "weight" which indicates the amount of data to be redistributed. Second, graph Voronoi diagram does not consider the "capacity" of each node, which in our problem, signifies the available storage space of sensor nodes.

III. Data Redistribution Problem

A. Network Model and Problem Formulation

In our model, there are some data generators generating large amount of sensory data, with total size much larger than their storage capacities. The sensory data are modeled as a sequence of raw data items, each of which has the same unit size. Each sensor node has limited storage capacity and can only hold finite amount of data items. Sensor nodes which collect more data than what they can store in their local storage, are the data generators and they have to redistribute/offload some of their data to other nodes that have available storage space. The objective of our data redistribution problem is to redistribute the data items from the data generators to other nodes to fully utilize the overall storage capacity of the sensor network, while minimizing the total energy consumption in the sensor network[2].

Given a general sensor network graph $G(V,E)$ where $V = \{1, 2, ..., N\}$ is the set of N nodes, and E is the set of edges. Two nodes are connected by an edge if they are within the transmission range of each other and thus can communicate directly. We assume sensor nodes are distributed uniformly at random in the deployment region since near-uniform node deployment is an easy and practical approach to provide full sensing coverage and connectivity, and commonly followed in sensor node placement. Let d_{ij} denote the shortest path distance (in terms of number of hops) between two sensor nodes i and j.

There are p data generators in the network. Without loss of generality, we assume that they are $\{1, 2, ..., p\}$. Data generator i is referred to as DG i. Let s_i denote the number of data items DG i needs to redistribute and m_i denote the available free storage space (in terms of number of data items) at sensor node $i \in \{p+1, p+2, ..., N\}$. If $s_i > 0$, then $m_i = 0$, meaning node i has a full storage space and thus can not store any more data items; in this case node i is a data generator. If $s_i = 0$, then node i can store m_i data items offloaded from other data generator nodes.

Each data generator redistributes one data item at a time. For our energy cost model, we use the number of hops to measure the energy consumption of redistributing the data item[3]. The *redistribution cost* for DG i, with s_i number of data items to redistribute, is the sum of the number of hops to redistribute all s_i data items. The *total redistribution cost* of the sensor network is defined as the sum of the redistribution cost of all the data generators. The goal of the problem is to redistribute the data items from the data generators into the network with minimum total redistribution cost. *Without loss of generality, we assume that the total size of the data items to be redistributed is less than or equal to the size of the total available storage space in the network.*

[2]Note that the goal is to minimize the total energy consumption in the network, and not to load balance the individual energy consumption at different sensor nodes. Our model focuses on the scenario where data redistribution occurs infrequently, and the total redistribution energy cost in the network is the optimization objective.

[3]We adopt the first order radio model [9] wherein for a k-bit data over distance l, the transmission energy $E_{Tx}(k,l) = E_{elec} * k + \epsilon_{amp} * k * l^2$, the receiving energy $E_{Rx}(k) = E_{elec} * k$, where E_{elec} and ϵ_{amp} are constant. With the uniform distribution of the sensor nodes, the average distance between any two neighboring sensor nodes could be assumed to be the same, and of unit length. Also since the data items are of equal sizes, the energy consumed to redistribute one data item over one hop is assumed to be the same throughout the network. Liu et al. [16] also assume that transmitting one packet over one hop consumes one unit of energy. Since the total energy cost equals energy cost at each hop times the number of hops, and the energy cost at each hop is assumed to be a constant, minimizing total energy cost is the same as minimizing number of hops.

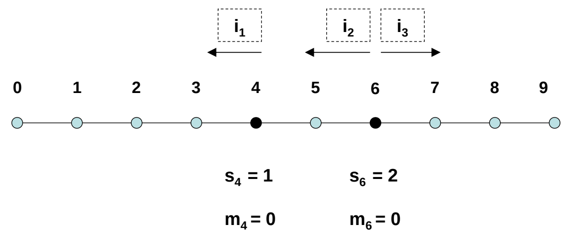

Fig. 2. Illustrating data redistribution problem with a linear network.

To formulate our problem, let I denote the set of data items to be redistributed in the whole network, and let $S(i)$, where $i \in I$, be data item i's data generator. A redistribution function is defined as $r : I \to V$, indicating data item $i \in I$ is redistributed to node $r(i) \in V$ via the shortest path between $S(i)$ and $r(i)$. Our goal is to find such a redistribution function r to minimize the total redistribution cost:

$$\sum_{i \in I} d_{S(i)r(i)}, \tag{1}$$

under the storage capacity constraint that the number of data items offloaded to node j is less than or equal to j's available storage capacity, i.e.,

$$|\{i | i \in I, r(i) = j\}| \le m_j, \qquad \text{for all } j \in V.$$

Below we give a simple example to illustrate the data redistribution problem under storage constraint.

EXAMPLE 1: Fig. 2 illustrates the data redistribution problem in a small linear sensor network. Each edge is of one hop. There are two data generators: node 4 has one data item, i_1, to redistribute; node 6 has two data items, i_2 and i_3, to redistribute. The storage capacity of all other nodes equals one data item each. The minimum cost solution is node 4 offloads i_1 to node 3, while node 6 offloads i_2 and i_3 to nodes 5 and 7, respectively. The total redistribution cost is 3. □

B. Minimum Cost Flow Problem

We show that our data redistribution problem is equivalent to the minimum cost flow problem. Recall that minimum cost flow problem [3, 22] is the following. Given a graph in which each edge has a capacity and a cost. Some nodes are supply nodes and some are demand nodes, and the total supply equals the total demand. The problem is to find flows from supply nodes to demand nodes with minimum cost such that the capacity constraint of each edge is satisfied.

Theorem 1: The data redistribution problem is equivalent to the minimum cost flow problem.

Proof: Given above general sensor network graph $G(V,E)$, let $V_1 = \{1, 2, ..., p\}$ be the set of p data generators, and V_2 be the rest $N - p$ sensor nodes, $V_1 \cup V_2 = V$. We reduce the data redistribution problem to the minimum cost flow problem by changing $G(V,E)$ into a new graph $G'(V', E')$ as follows (shown in Fig. 3).

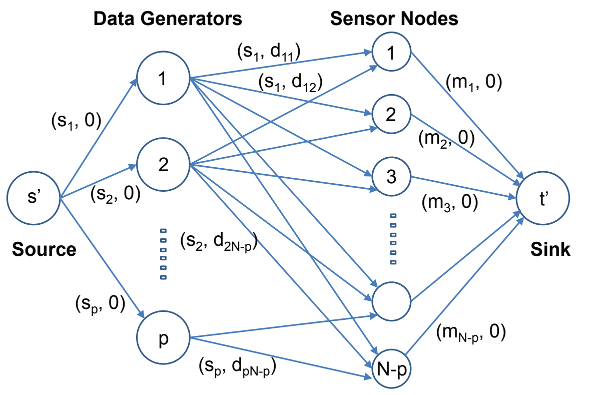

Fig. 3. Data redistribution problem is equivalent to minimum cost flow problem.

1. $V' = V \cup \{s'\} \cup \{t'\}$, where s' is the new source node, and t' is the new sink node.
2. $E' = \{(i,j) : i \in V_1 \text{ and } j \in V_2\} \cup \{(s',i) : i \in V_1\} \cup \{(j,t') : j \in V_2\}$.
3. For each edge (i,j), set its capacity as s_i, and its cost as d_{ij}, which is the shortest distance between DG i and sensor node j in original graph G(V,E).
4. For each edge (s',i), set its capacity as s_i and its cost as 0. For each edge (j,t'), set its capacity as m_j and its cost as 0.
5. Set both the supply at s' and the demand at t' as $\sum_{i=1}^{p} s_i$. The supply of other nodes in V' is set as 0.

Now a valid flow of amount $\sum_{i=1}^{p} s_i$ from s' to t' includes s_1 amount on edge $s'1$, s_2 amount on $s'2$, ..., and s_p amount on $s'p$. This is actually the maximum possible flow and it exists due to the assumption of $\sum s_i \le \sum m_j$. Therefore solving the minimum cost flow problem on $G'(V',E')$ gives the minimum redistribution cost in our data redistribution problem in $G(V,E)$.

■

The minimum cost flow problem can be solved efficiently in polynomial time using well-known algorithms [2, 6–8, 10, 21, 28]. In this paper, we use the algorithm and implementation by Goldberg [6, 7] due to its practical nature. This algorithm has the time complexity of $O(N^2 M log(NC))$, where N, M, and C are the number of nodes, the number of edges, and the maximum capacity of an edge in graph G'. In our case, $C = \max_i\{s_i\}$. If C is not very large, Goldberg's algorithm is feasible for the minimum cost flow problem. Otherwise, other strong polynomial algorithms [21] ($O((MlogN)(M + NlogN))$), [28] ($O(M^4)$) can be used.

IV. Potential-based Distributed Algorithm (PDA)

To introduce our distributed algorithm, let's start again with the Example 1 depicted in Fig. 2. The minimum cost optimal solution is that node 4 sends its one data item to node 3, while node 6 sends one data item to node 5 and the other one to node 7. However, in a distributed environment, since both node 3 and node 5 have the same distance to node 4, node 4 could offload its data to node 5, resulting in non-optimal solution. In this section, we show how data redistribution is performed using the concept of *potential*. We first introduce the basic potential field model. Then, we present our potential-based distributed algorithm called PDA, followed by the discussion of its convergence and performance.

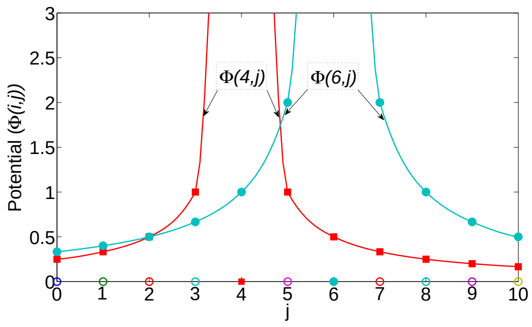

Fig. 4. Potential field of the sensor network in Example1.

A. Potential Field Model

We study data redistribution using an analogy that the whole sensor network is an electric potential field, wherein each data generator is an electric charge. For DG i with s_i data items to offload, it has a positive electric charge of s_i. For arbitrary sensor node j, its potential due to DG i, denoted as $\phi(i,j)$, is equal to $k_0 \frac{s_i}{d_{ij}}$, where k_0 is a constant and d_{ij} is the distance between DG i and j (we omit k_0 for the rest of the paper). According to superposition principle [12], the total potential field of the whole sensor network is the linear superposition of all individual fields of the data generators. For arbitrary node j, denote its total potential as $\phi(j)$, we get:

$$\phi(j) = \sum_{i=1}^{p} \phi(i,j) = \sum_{i=1}^{p} \frac{s_i}{d_{ij}}. \tag{2}$$

Fig. 4 shows the individual potentials at different sensor nodes due to the data generators (nodes 4 and 6) for the linear sensor network depicted in Fig. 2. Here, $s_4 = 1$, and $s_6 = 2$. The potential values decrease symmetrically for sensor nodes in both directions as their distance from data generator increases. Note that $\phi(5)$ $(= \phi(4,5)+\phi(6,5) = 1+2 = 3)$ is the maximum among all sensor nodes, suggesting that contention is highest at this node. Also note that at node 1, the potential due to data generator node 6 is higher than that due to data generator node 4, ie. $\phi(6,1)\left(=\frac{2}{5}\right) > \phi(4,1)\left(=\frac{1}{3}\right)$, even though node 1 is located closer to node 4. This is due to the fact that the amount of data that needs to be redistributed also plays a major role in defining the potential values. The potential values serve an important role in the design (and performance) of our distributed data redistribution algorithm. Intuitively, a sensor node prefers to commit storage space to the data generator whose potential at the sensor node is the highest. In addition, the total potential of a sensor node is the key towards informing the data generators of the level of contention at a sensor node. Thus the data generator node 4 in Example 1 could look at the total potential at nodes 3 and

5 and could decide to offload its data item to node 3 due to its lower total potential.

B. Potential-based distributed algorithm (PDA)

The PDA takes place in iterations. Each iteration consists of the following three stages:

1. **Advertisement Stage.** For DG i that has data items to offload, it floods an advertisement message to the network with its ID and number of data items to offload (s_i)[4]. An integer (initialized as 0) is included in the advertisement message and incremented every time the message is forwarded. This information is used to capture the distance between any sensor node and DG i.
2. **Storage Commitment Stage.** For each sensor node j with available storage space $m_j > 0$, on receiving advertisement message from DG i, it performs the following steps (In addition, each node j forwards the advertisement message, the first time it receives it.):
   - A. Computes its potential value due to the DG i, $\phi(i,j) = \frac{s_i}{d_{ij}}$. $\phi(i,j) = 0$ if j did not receive DG i's advertisement message. It also computes its total potential $\phi(j)$ after receiving all the advertisement messages.
   - B1. Finds the data generator that gives the maximum potential value. Ties are broken randomly. Suppose such data generator is DG k where $k = \text{argmax}_{1 \le i \le p} \phi(i,j)$, j *commits* one unit of storage space to DG k and updates $m_j = m_j - 1$.
   - B2. If $m_j > 0$, j still has free storage space to commit, it updates $s_k = s_k - 1$, $\phi(k,j) = \frac{s_k}{d_{kj}}$, and goes back to Step B1. Otherwise, it has committed all its storage and goes to Step C.
   - C. Sends a message to each data generator (DG i) to which it has committed storage, along with the number of storage space committed (c_{ij}), its current total potential $\phi(j)$, as well as d_{ij}. Note d_{ij} has been obtained in Stage 1.
3. **Data Offloading Stage.** We denote the set of sensor nodes who commit storage to DG i as $\mathcal{C}_i$. After receiving all the commitment messages from $\mathcal{C}_i$, each DG i performs the following computations:
   - A. Compares the total number of received commitment, $\sum_{j \in \mathcal{C}_i} c_{ij}$, with its current number of data items to offload, s_i. If $\sum_{j \in \mathcal{C}_i} c_{ij} > s_i$, it goes to Step B1 below. Otherwise, DG i can completely satisfy all the commitment and thus sends to each committed sensor node the amount of data it committed to store for DG i. After this, it updates $s_i = s_i - \sum_{j \in \mathcal{C}_i} c_{ij}$. If $s_i > 0$, DG i still has data to offload, it starts another iteration and goes back to Stage 1 for advertisement.
   - B1. In this step, DG i needs to decide how many data items to offload to which of the committed sensor nodes. To do that, DG i decides to offload one data item to the closest sensor node, say node k, among all the committed sensors $j \in \mathcal{C}_i$. If there are multiple closest nodes, break the tie by choosing the node with the least total potential. Then, DG i updates $s_i = s_i - 1$, and $c_{ik} = c_{ik} - 1$. If $c_{ik} = 0$ (the data generator has decided to offload as much data to node k as committed by node k during storage commitment stage), remove k from the set $\mathcal{C}_i$.
   - B2. If DG i still has data to redistribute ($s_i > 0$), it recomputes $\phi(j) = \phi(j) - \frac{1}{d_{ij}}$ for all $j \in \mathcal{C}_i$, and goes back to Step B1 to find the next sensor node to offload one data item. Otherwise, DG i goes to Step C.
   - C. DG i offloads all the data items to sensor nodes according to above calculation.

[4]We adopt *pure flooding*, wherein each node only broadcasts the advertisement message the first time it receives it. Therefore, the message complexity in advertisement stage is only $O(N)$.

PDA stops when all the data generators have offloaded their data items. In each iteration, each sensor commits all its storage space. However, it could be the case that the sensor's commitment is not fully utilized by data generators. At the end of an iteration, if a node receives less offloaded data than what it had committed, the node is free to commit its remaining available storage in the next iteration. Note that either sensor nodes that no longer have storage spaces available or data generators that no longer have data items to offload do not actively participate in the next iteration, other than forwarding the advertisement message.

Discussion of PDA. It is easy to check that PDA solves the Example 1 optimally in just one iteration. PDA takes place in iterations, thus some synchronization is needed. However, PDA is a fully distributed, highly scalable, and efficient data distributed mechanism, with the following characteristics.

- PDA is an online distributed algorithm and applicable to environments where data generation occurs dynamically. It does not require the data generators to communicate with each other for redistribution contention resolution. The contention is resolved during the storage commitment stage by the sensor nodes.
- In PDA, data generators do not need to have the knowledge of the remaining storage capacity of other sensor nodes. In addition, the sensor nodes do not have the knowledge of the data generator's redistribution needs.
- It is also adaptable to network dynamics such as dynamic data generating and node failures.

Note that in the storage commitment stage, each sensor node will commit all its free storage space, even when the total number of data items advertised in the received advertisement messages is less than its storage space size. This *over-commitment* happens only when $\sum_{i=1}^{p} s_i < m_j$ for some sensor node j. If that happens, the current iteration becomes the last iteration of the algorithm as all data generators can offload all their data in the current iteration. Thus over-commitment does not incur much computational overhead in PDA.

PDA is more applicable in challenging scenarios where the number of data items and number of data generators are

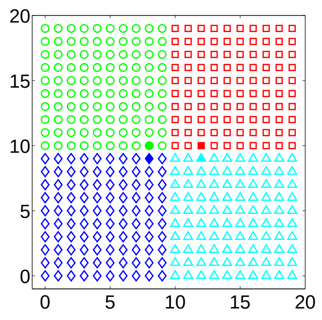
Fig. 5. Optimal (redistribution cost = 3160).

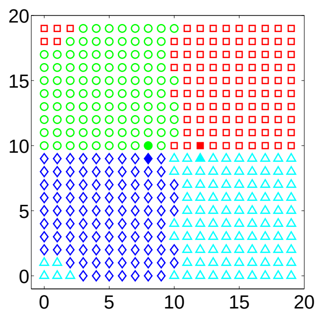
Fig. 6. PDA (redistribution cost = 3205).

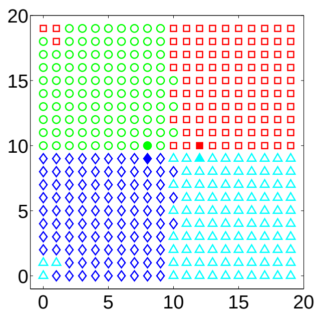
Fig. 7. Cooperative (redistribution cost = 3200).

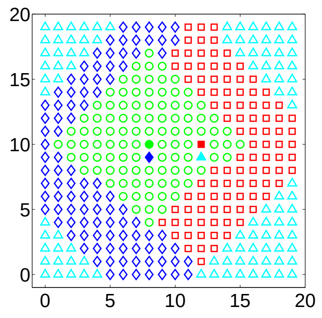
Fig. 8. Greedy (redistribution cost = 3524).

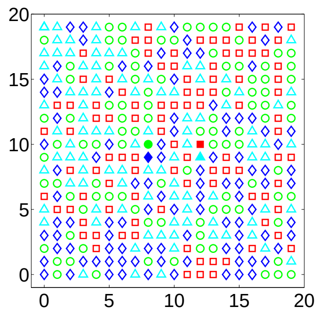
Fig. 9. Random (redistribution cost = 5235).

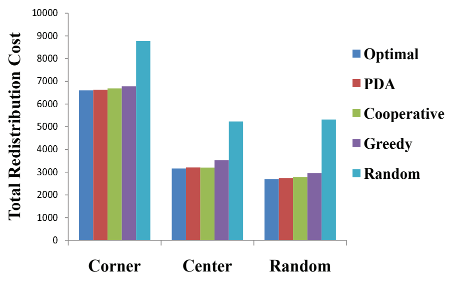

Fig. 10. Performance with different DG locations.

TABLE I
TIME COMPLEXITY COMPARISON OF ALL DATA REDISTRIBUTION ALGORITHMS.

Optimal	Greedy	Cooperative	Random	PDA
$O(N^2Mlog(NC))$	$O((p+\bar{m})N^2)$	$O((p+\bar{m})N^2)$	$O(\bar{m}N^2)$	$O(\bar{m}N^3)$

large. In sparse networks (where number of data generators and/or the number of data items are less), the PDA algorithm could be modified to reduce the complexity by sending out advertisements to a few hops only. Or a simple greedy heuristic could be used instead of PDA.

C. Performance and convergence analysis of PDA

Below we provide convergence and performance analysis of PDA, including its message and time complexity analysis.

Theorem 2: The PDA will stop in at most p iterations, where p is the number of data generators in the network.

Proof: We first show that in each iteration, at least one data generator would receive commitment more than its number of data items to be offloaded, by way of contradiction. Assume that in the first iteration of PDA, none of the data generator receives more commitment than number of its data items. That is,

$$\sum_j c_{ij} < s_i, \quad \forall i \in \{1, 2, ..., p\}.$$

Sum up among all the data generators, we get

$$\sum_i (\sum_j c_{ij}) < \sum_i s_i,$$

which implies

$$\sum_i \sum_j c_{ij} < \sum_i s_i.$$

Since all sensor nodes commit all their storage space disjointly to different data generators,

$$\sum_i \sum_j c_{ij} = \sum_j m_j,$$

which implies

$$\sum_j m_j < \sum_i s_i.$$

This contradicts with the assumption that the total size of the data items to be redistributed of all the data generators is less than or equal to the size of the total available storage space in the network. So at least one data generator finishes offloading in the first iteration.

In subsequent iterations, total data to be offloaded $(\sum_i s_i)$ and total available storage $(\sum_j m_j)$ get decremented by exactly the same amount. Therefore, the result holds for other iterations as well. Thus PDA takes at most p iterations. ∎

Message Complexity. We calculate the number of transmissions in PDA. Each iteration there are at most p data generators broadcasting advertisement messages, reaching all the

N sensor nodes. There are at most p iterations from Theorem 2. So the total number of transmissions of advertisement messages is $O(p^2N)$. In each storage commitment stage, a sensor node sends at most p commitment messages. The number of hops from any sensor node in the network to any data generator node is at most N. So the total number of transmissions of commitment messages is O(p^2N^2). For the data offloading, there are total $\sum_{i=1}^{p} s_i \le N\bar{m}$ data items to offload, each travels at most N hops, resulting in total $O(N^2\bar{m})$ offloading transmissions, where $\bar{m}$ is the average storage capacity of each sensor node. So the total number of transmissions (or message overhead) in PDA[5] is $O((p^2 + \bar{m})N^2)$.

Time Complexity. In the storage commitment stage of each iteration, each sensor node makes an average $\bar{m}$ commitments, each of which is towards one of (at most) p data generators. So the total N sensors incur $O(N\bar{m}p)$ computations. There are at most p iterations. The total number of commitment computations is thus $O(N\bar{m}p^2)$. For the data offloading, to offload one of its data items, each data generator chooses one sensor node, which takes $O(N)$ computations. There are at most $N\bar{m}$ data items. So the total number of computations in data offloading is $O(N^2\bar{m})$. Therefore the total time complexity of PDA is $O(N\bar{m}p^2 + N^2\bar{m})$, which is of the order $O(N^3\bar{m})$, since p can be at most of order $O(N)$.

D. Centralized Heuristics

We design a set of centralized and intuitive heuristics, namely random algorithm (Random), greedy algorithm (Greedy), and cooperative algorithm (Cooperative), and later compare their performances with PDA and Optimal.

In Random, each data generator randomly selects a sensor node to offload its data items. Its time complexity is $O(\sum_{i=1}^{p} s_iN) = O(\bar{m}N^2)$. It is the most efficient heuristic in terms of time complexity.

In Greedy, data generators take turn to redistribute the data in the ascending order of their IDs. For each data generator, it performs a BFS ordering of all other nodes ($O(N^2)$), and offloads all the data items to the closest unoccupied sensor nodes ($O(s_iN)$ for DG i, and tie is broken randomly). So the time complexity of Greedy is $O(pN^2 + \sum_{i=1}^{p} s_iN) = O(pN^2 + N\bar{m}N) = O((p + \bar{m})N^2)$. Note the first equality is due to $\sum_{i=1}^{p} s_i \le N\bar{m}$.

Cooperative takes place in rounds. In each round, similar to the Greedy, the data generators take turn to redistribute the data in the ascending order of their IDs. However, unlike the Greedy, each data generator only offloads one data item at a time to its closest unoccupied sensor node (ties are broken randomly). The time complexity is the same as that of Greedy, i.e., $O((p + \bar{m})N^2)$, however this heuristic tends to perform much better than Greedy as we show in Section V.

Table I shows the time complexity comparison of different data redistribution algorithms. In terms of time complexity, Random $<$ Greedy, Cooperative $<$ PDA $<$ Optimal. Note that, the centralized heuristics (as well as Optimal) are not suitable for distributed environments, and are considered mainly for performance comparisons with PDA.

[5]Note that we do not account for this message complexity in the redistribution cost. Nevertheless, we compare the message overhead of PDA with that of EnviroStore in Section V.

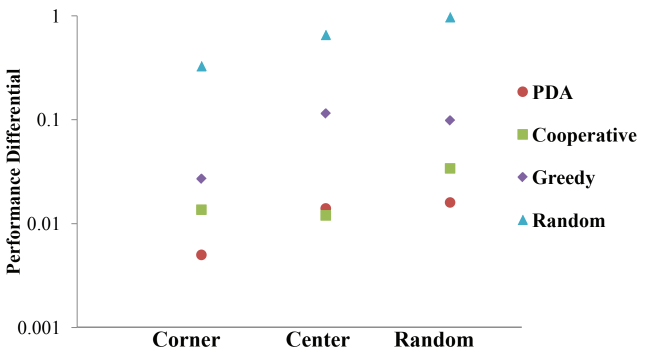

Fig. 11. Performance percentage differential under different scenarios.

V. Performance Evaluation

In this section, we first compare the performance of PDA and other centralized heuristics with that of the optimal solution for the minimum cost flow problem. We then compare PDA with EnviroStore, an existing data redistribution scheme in sensor networks.

A. Comparison among Optimal, PDA, and Other Heuristics

We first visually compare the performances of different algorithms. For this purpose, we assume each sensor node has one unit storage space and adopt a sensor network with grid-like topology (note that our proposed algorithms are applicable to all topologies). We then compare different algorithms under various network scenarios wherein data generators' locations are varied appropriately. To study the scalability of each algorithm, we vary the number of data generators and number of data items to redistribute in a large scale sensor network (10,000 nodes), and compare the performances of different algorithms. In all cases, the transmission range of the sensor is one unit, the length of each grid edge.

Visual Performance Comparison. We deploy 400 sensor nodes evenly on a 20×20 grid network. Fig. 5-9 show the visual comparison of the following algorithms: Optimal, PDA, Cooperative, Greedy, and Random, with the total redistribution cost as indicated. There are four data generators in the network, located at (8, 10), (12, 10), (8, 9), (12, 9). Each data generator has 99 data items to offload. The four different filled shapes correspond to location of the four data generators. The unfilled shapes correspond to the offloading area of their respective data generators. We observe that Cooperative and PDA both perform close to Optimal. However, the performance of Greedy and Random algorithms is far from optimal. This is visibly apparent from the structure of offloading areas for different data generators in the optimal solution. Both PDA and Cooperative are able to successfully estimate this inherent structure. Note that the difference in performance between PDA and Cooperative is not large in this case. However, as we shall show later in Fig. 12, when the size of the problem is sufficiently large, PDA outperforms Cooperative. In addition, note that Cooperative is an intuitive but centralized heuristic, and thus is not suitable for practical deployment.

Performance Comparison Under Various Scenarios. We use the same parameters as in the above visual performance comparison, and compare different algorithms under the following scenarios: 1) all data generators are located at one corner of the network; 2) all data generators are located at the center of the network; 3) all data generators are randomly placed in the network. Fig. 10 compares the total redistribution cost of each algorithm under these scenarios. We observe that, all the algorithms incur the largest redistribution cost when all the data generators are at one corner, followed by when all the data generators are at the center, and the cost is the smallest when data generators are randomly placed in the network. This is as expected, since with data generators in one corner, most of the data items have to be offloaded to distant nodes, compared with the other two scenarios. Similarly, the random location results in less total redistribution cost compared to the center scenario. In all the cases, the performance trend observed is given by Optimal $>$ PDA $>$ Cooperative $>$ Greedy $>$ Random.

Performance Percentage Differential (PPD). Next, we explore how different algorithms perform compared to Optimal under different scenarios. We calculate the *performance percentage differential* of each algorithm, which is defined as the difference of the cost between the algorithm and Optimal, divided by the cost of Optimal. Fig. 11 shows the PPD of Greedy, Random, Cooperative, and PDA algorithms. We observe that the performance difference of PDA is within 5% of the optimal performance in all the scenarios. Also note that as the location of data generator is changed from corner to random, the optimal redistribution cost decreases, but the PPD increases in almost all cases. This is because the optimal solution seems to have a better structured offloading area for each data generator in center and random scenarios, which is difficult to estimate using either PDA or the heuristics [6].

Varying Number of Data Generators and Number of Data Items. Below we study the performance comparison by varying number of data generators and data items to be redistributed. We consider a 100×100 network, with 10000 sensor nodes each with unit storage capacity. We vary the number of data generators from 20, 40, 60 to 80 and total number of data items of each data generator from 50, 70 to 90. The data generators are randomly placed in the network. From Fig. 12, we observe that PDA performs comparable to Optimal and Greedy algorithm performs worse than Cooperative. The difference between the redistribution algorithms is seen clearly when there are more number of data generators with larger amount of data items to be redistributed. As the number of data generators and the total data items increase, the PPD of the centralized heuristics increase significantly compared with PDA. Therefore, PDA is suitable for large scale data intensive

[6]Due to space limitations, we omit the visual performance comparison for corner and random scenarios.

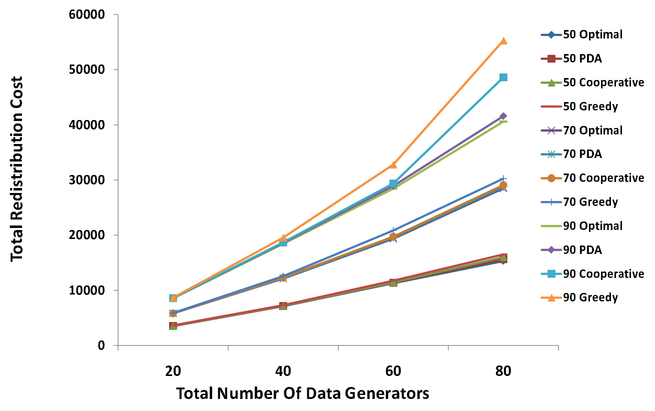

Fig. 12. Varying number of data generators and data items in 100×100 network.

sensor networks, with heavy data redistribution requirements.

We observed with our simulations that the proposed Potential Field Based data redistribution algorithm performs comparable to the optimal. In all the scenarios considered, the difference between the optimal performance and the performance obtained by PDA is less than 5%.

B. *Comparison between PDA and EnviroStore[18]*

We implement PDA using TinyOs 2.1, and compare it with EnviroStore using TOSSIM simulator[7]. We first provide an overview of EnviroStore.

Overview of EnviroStore. EnviroStore is a cooperative storage system designed for disconnected sensor networks, where the sensed data is stored inside the network until it is collected by human operator or data mule. EnviroStore addresses how to fully utilize the storage of the network in an energy-efficient way. To do this, each node monitors its own remaining storage and exchanges this information periodically with its one-hop neighbors. When its remaining storage is less than a particular threshold and the imbalance (or difference) between the average remaining storage of its neighbors and its own remaining storage is larger than another threshold, the node begins to offload some amount of data to one of the under-loaded neighbors. However, if several nodes simultaneously offload data to the same node, this node will be overloaded in storage and will offload data back to its neighbors, causing unnecessary energy and bandwidth consumption. To prevent this so called data ping-pong phenomenon, in EnviroStore, the amount of data offloaded is bounded using appropriate thresholds based upon the imbalance value mentioned above. EnviroStore also provides reliable unicast for nodes to transfer data. The simulations in [18] show significant improvement in the amount of data collected using EnviroStore.

Routing Support in PDA. One difference between PDA and EnviroStore is that in PDA, nodes rely upon routing tables for storage commitment and data offloading. However, the

[7]EnviroStore is implemented in TinyOS 1.x. The major difference between 1.x and 2.x is that 2.x includes layer-2 source addresses in the packets while 1.x does not, which does not affect the algorithm comparison in this paper.

necessary routing information can be readily obtained in advertisement and commitment stages of PDA, without costing additional overhead messages. In advertisement stage, when a sensor receives the advertisement message from a neighbor, it records the neighbor as the next hop to the data generator of the advertisement message. This information is used in storage commitment stage to route the commitment messages back to the data generators. Meanwhile, in storage commitment stage, a node also constructs routing information for the committing sensor nodes, and this information is used for data delivery in data offloading stage. In both advertisement and commitment stages, a TTL field is used in the message to record the number of hops between nodes. Like EnviroStore, reliable communication is also implemented in PDA, in both commitment and data offloading stages.

Simulation Setup. We adopt the grid-like sensor deployment in [18], with 36 nodes placed in a grid manner; the only difference being that all the nodes are connected in the deployment, since we focus on in-network data redistribution in this paper. In the deployment, each sensor can communicate directly with its neighbors only. Among the 36 nodes, two nodes are configured as data generators, which periodically generate data and create input for both EnviroStore and PDA. For EnviroStore simulation, we adopt all the default threshold values in [18], i.e., the remaining storage threshold is 0.95S, imbalance threshold (between the average remaining storage of its neighbors and its own remaining storage) is 0.05S, node advertisement threshold is 0.01S (S is the total storage of a node, which is 16KB). The size of each data packet is set as 22 bytes in both PDA and EnviroStore simulations[8]. The data generating rate is set as 64 bytes/sec in both schemes.

Comparison of Redistribution Cost. In this part, we compare the total redistribution cost (using Equation 1) of EnviroStore and PDA at various values of simulation running time. The time for each iteration in PDA is set to 80 seconds. Fig. 13 shows that the total redistribution cost of PDA is much smaller than that of EnviroStore for most values of the simulation time, indicating PDA is more energy-efficient than EnviroStore in terms of the solution quality. There are mainly two reasons for this. First, in EnviroStore, each sensor (including the data generator) tends to redistribute data to far-away nodes even when the nearby nodes have not exhausted their storage capacities. In addition, the sensor nodes to which the data is offloaded, can again redistribute this data to other nodes. Such behavior leads to ping-pong phenomena and could possibly result in data redistribution loops. While in PDA, the DGs offload all of their data to nearby nodes, and these sensor nodes do not redistribute the data items avoiding ping-pong phenomena and loops. Fig. 14 depicts this performance comparison in detail. Second, EnviroStore only requires each node to talk with its one-hop neighbors. Whereas in PDA, the advertisement and storage commitment phases enable

[8]In [18], the size of each data is 32 bytes, which requires two data messages to transmit each data (one data message can transmit 29 bytes of data). This results in a waste of message space and additional energy consumption for data delivery. Thus, we set the data size to 22 bytes so that it can be delivered using one data message (plus 7 bytes header).

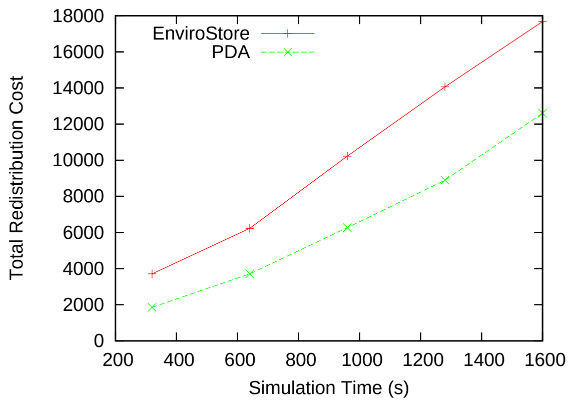

Fig. 13. Comparison of total redistribution cost between EnviroStore and PDA.

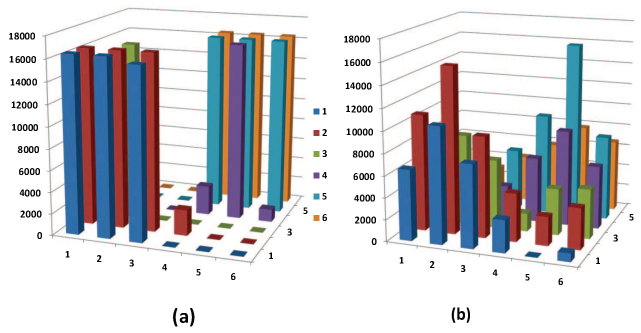

Fig. 14. 3D comparison of data redistribution between EnviroStore and PDA. (a) PDA, (b) EnviroStore. Here, the z-axis represents the amount of data redistributed to sensor nodes at different grid locations. When using PDA, most of the data gets redistributed to nodes near the DG locations, unlike in EnviroStore.

the effective communication between data generators and all other sensor nodes leading to a more energy-efficient data redistribution. Therefore, PDA seems to be able to utilize the network storage better than Envirostore. In addition, PDA can naturally eliminate data ping-pong problem, reducing the unnecessary energy consumption incurred in EnviroStore.

Comparison of Message Overhead. Above comparison only focuses on the quality of the solution achieved by PDA and EnviroStore. Here, we study the message overhead of both schemes. The overhead messages in PDA include the advertisement messages and the storage commitment messages while the overhead messages in EnviroStore includes periodic advertisement messages sharing remaining storages among nodes. From the Fig. 15 we observe that the message overhead of PDA is less than that of EnviroStore. The reason is that PDA does not perform periodic advertisement messages, instead PDA incurs advertisement at the beginning of each iteration. Therefore, a decrease in the time period of one iteration (for eg. from 320 to 160 seconds in Fig 15) leads to an increase in the occurrence frequency of advertisement stages, resulting in larger overhead over a period of time T. If the time period of one iteration is reduced further, at some small value the overhead of PDA would become comparable

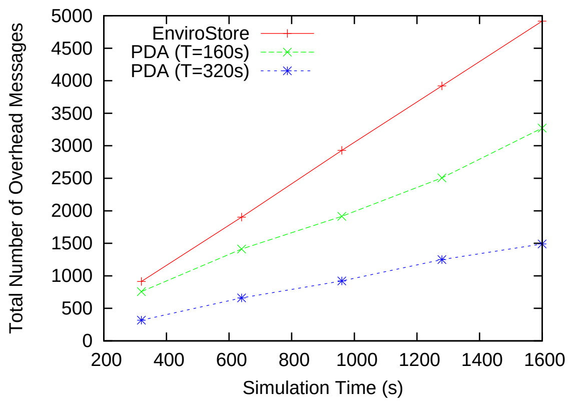

Fig. 15. Comparison of total overhead messages between EnviroStore and PDA.

to that of EnviroStore.

VI. Conclusion and Future Work

In this paper, we study the data redistribution problem in sensor networks. Our results are two-fold. First, we show that the data redistribution problem is equivalent to the minimum cost flow problem, which can be solved optimally in a centralized manner. Second, for a distributed algorithm, we have applied the idea of electrostatic potential field to develop a distributed data redistribution mechanism. Through simulations, we show that our distributed algorithm performs very close to the optimal centralized solution. In all the scenarios considered, the difference between the optimal performance and the performance obtained by PDA is less than 5%. Using TOSSIM simulator, we show that our proposed distributed algorithm performs better than the existing data redistribution technique (EnviroStore) in terms of both the solution quality and overhead messages.

In future, we plan to explore the effect of spatio-temporal correlations among the data generated in sensor network on the performance of the proposed redistribution algorithm. Also, we have only considered data redistribution due to storage overflow. However, data redistribution would also be needed in case of energy depletion at the sensor nodes, which we plan to explore in future. Energy depletion triggered redistribution would lead to multiple redistribution instances for the same data item over time, and network lifetime would be an appropriate metric to optimize in that case.

VII. Acknowledgement

We would like to thank Dr. Bayram Yildirim for helpful discussions. We also thank Ashok Madhvesh for help with Matlab.

References

[1] G. Aathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. In *Proc. of IPSN 2006*.

[2] R. Ahuja, A. V. Goldberg, J. Orlin, and R. E. Tarjan. Finding minimum-cost flows by double scaling. *Mathematical Programming*, 53:243–266, 1992.

[3] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[4] Martin Erwig and Fernuniversitat Hagen. The graph voronoi diagram with applications. *Networks*, 36:156–163, 2000.

[5] J. Gao, L. Guibas, N. Milosavljevic, and D. Zhou. Distributed resource management and matching in sensor networks. In *Proc. of IPSN 2009*.

[6] A. V. Goldberg. Andrew goldberg's network optimization library. http://www.avglab.com/andrew/soft.html.

[7] A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22(1):1–29, 1997.

[8] A. V. Goldberg and R. E. Tarjan. Solving minimum-cost flow algorithms by successive approximation. *Mathematics of Operations Research*, 15(3):430–466, 1990.

[9] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of HICSS 2000*.

[10] Bruce Hoppe and E. Tardos. The quickest transshipment problem. *Math. Oper. Res.*, 25(1):36–62, 2000.

[11] K. R. Chowdhury I. F. Akyildiz, T. Melodia. A survey on wireless multimedia sensor networks. *Computer Networks*, 51(4):921–960, 2007.

[12] J. D. Jackson. *Classical Eletrodynamics*. John Wiley and Sons, third edition, 1999.

[13] Samir Khuller and Yoo ah Kim. Algorithms for data migration with cloning. In *SIAM Journal on Computing*, pages 27–36. ACM Press, 2003.

[14] V. Lenders, M. May, and B. Plattner. Density-based anycast: A robust routing strategy for wireless ad hoc networks. *IEEE/ACM Transactions on Networking*, 16:852–863, 2008.

[15] S. Li, Y. Liu, and X. Li. Capacity of large scale wireless networks under gaussian channel model. In *Proc. of MOBICOM 2008*.

[16] Changlei Liu and Guohong Cao. Distributed monitoring and aggregation in wireless sensor networks. In *Proc. of Infocom 2010*.

[17] L. Luo, Q. Cao, C. Huang, L. Wang, T. Abdelzaher, and J. Stankovic. Design, implementation, and evaluation of enviromic: A storage-centric auio sensor network. *ACM Transactions on Sensor Networks*, 5(3):1–35, 2009.

[18] L. Luo, C. Huang, T. Abdelzaher, and J. Stankovic. Envirostore: A cooperative storage system for disconnected operation in sensor networks. In *Proc. of INFOCOM 2007*.

[19] K. Martinez, R. Ong, and J.K. Hart. Glacsweb: a sensor network for hostile environments. In *Proc. of SECON 2004*.

[20] N. T. Nguyen, A.-I. A. Wang, P. Reiher, and G. Kuenning. Electricfield-based routing: A reliable framework for routing in manets. *ACM Mobile Computing and Communications Review*, 8(2):35–49, 2004.

[21] J. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2):338–466, 1990.

[22] C.H. Papadimitriou and K. Steiglitz. Combinatorial optimization: Algorithms and complexities. *Prentice Hall*, 1982.

[23] A. Pinar and B. Hendrickson. Interprocessor communication with limited memory. *IEEE Transactions on Parallel and Distributed Systems*, 15:606–616, 2004.

[24] Rahul C. Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *Proc. of SNPA 2003*.

[25] Stephen F. Siegel and Andrew R. Siegel. Madre: The memory-aware data redistribution engine. In *Proc. of PVM/MPI 2008*.

[26] S. Soro and W. Heinzelman. A survey of visual sensor networks. *Advances in Multimedia*, 2009.

[27] Affan A. Syed, Wei Ye, and John Heidemann. T-lohi: A new class of mac protocols for underwater acoustic sensor networks. In *Proc. of INFOCOM 2008*.

[28] E. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, 1985.

[29] S. Toumpis. Mother nature knows best: A survey of recent results on wireless networks based on analogies with physics. *Computer Networks*, 52:360–383, 2008.

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

[31] Geoff Werner-Allen, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proc. of OSDI 2006*.