Energy-Efficient Data Redistribution in Sensor Networks

BIN TANG and NEERAJ JAGGI
Wichita State University
HAIJIE WU
Texas A&M University
and
ROHINI KURKAL
Wichita State University

We address the *energy-efficient data redistribution problem* in data-intensive sensor networks (DISNs). In a DISN, a large volume of data gets generated, which is first stored in the network and is later collected for further analysis when the next uploading opportunity arises. The key concern in DISNs is to be able to redistribute the data from data-generating nodes into the network, under limited storage and energy constraints at the sensor nodes. We formulate the data redistribution problem where the objective is to minimize the total energy consumption during this process, while guaranteeing full utilization of the distributed storage capacity in the DISNs. We show that the problem is APX-hard for arbitrary data sizes; therefore, a polynomial time approximation algorithm is unlikely. For unit data sizes, we show that the problem is equivalent to the 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. Thus we design a distributed algorithm for the data redistribution problem which performs very close to the optimal, and compare its performance with various intuitive heuristics. The distributed 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 study the convergence and performance of the proposed algorithm and demonstrate its near-optimal performance and scalability under various network scenarios. In addition, we implement the distributed algorithm in TinyOS, 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 message overhead. Finally, we extend the proposed algorithm to avoid disproportionate energy consumption at different sensor nodes without compromising on the solution quality.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed Networks, Wireless Communication*; H.3.4 [**Information**

Authors' address: Bin Tang, Neeraj Jaggi, and Rohini Kurkal, Department of Electrical Engineering and Computer Science, Wichita State University, Wichita, KS 67260, email: bintang@cs.wichita.edu, neeraj.jaggi@wichita.edu, rxkurkal@wichita.edu; Haijie Wu, Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843, email: bylike@neo.tamu.edu.
Preliminary version appeared in Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), 2010.

Storage and Retrieval]: Systems and Software—*Distributed Systems*; F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Non-numerical Algorithms and Problems*

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Data Redistribution, Energy Efficiency, Sensor Networks

1. 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 ecological monitoring [Luo et al. 2007], visual and acoustic sensor networks [Soro and Heinzelman 2009; Luo et al. 2009], underwater or ocean seismic sensor networks [Vasilescu et al. 2005; Syed et al. 2008; Li et al. 2008] and geophysical monitoring [Werner-Allen et al. 2006; Martinez et al. 2004].

Many of the above emerging sensor network applications are deployed in challenging environments, such as in remote and inaccessible regions, while monitoring the environments for long time periods. Therefore sensor network could remain disconnected from the base station or even operate without a base station in the field for substantially large periods of time. Data generated inside the network is uploaded to the distant base station via different means and uploading opportunities, instead of multi-hop wireless communication usually adopted when the base station is nearby. The uploading opportunities could be periodic visit by human operators or data mules [Shah et al. 2003], or transmission to the base station through low-rate satellite link [Mathioudakis et al. 2007]. In a challenging environment, such uploading opportunities would be unpredictable and rare, making network connectivity to the base station inherently intermittent. Between two uploading opportunities, the generated data have to be stored inside the network. Therefore, one of the main functionalities of the sensor networks in these applications is to store the large amounts of data generated in the network before the next uploading opportunity arises.

Despite the advances in large lower-power flash memory such as parallel NAND flash technology [Mathur et al. 2006], storage is still a serious resource constraint in DISNs. According to [Luo et al. 2009], an acoustic sensor that has a 1GB flash memory and is designed to sample the entire audible spectrum will run out of its storage in just seven hours. Meanwhile, scientific data sensed from the physical environment is rich with information and its accurate analysis provides us with a closeup view of the physical environment. Therefore, data should be treated as the first class citizen in any sensor network scientific applications; and its preservation is of great importance.

Due to the uneven data generation (e.g., sensors close to the event of interest may collect data more frequently than nodes far away), some nodes collect more data and run out of their storage space more quickly than others and can not store

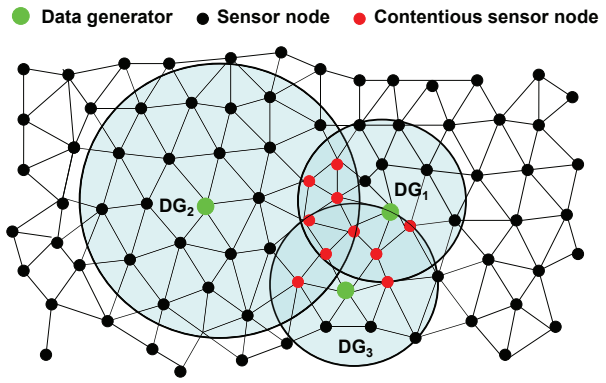

Fig. 1. Data redistribution problem with three data generators DG_1, DG_2, and DG_3. Each shaded circle represents each data generator's offloading area (the set of sensor nodes that are likely to store the offloaded data from data generators). Redistribution contention arises when data generators have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

more newly generated data. Such sensor nodes are referred to as *data generators*.[1] Therefore, when the storage capacity of a sensor is reached, the data needs to be redistributed/offloaded to other nodes with free storage space to avoid data loss. However, 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 comprised 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 sensor nodes, while minimizing total energy consumption incurred by the data redistribution.[2]

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 that store the offloaded data comprise the *offloading area* of data generators and are represented as shaded 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, *re-*

[1]In the remainder of this paper, we refer to non-data generator sensor nodes simply as sensor nodes.

[2]Note that in this paper, we do not consider data retrieval (and the cost incurred) and assume that the retrieval is done by data mules, human operators, or low-rate satellite link, using techniques such as those proposed in [Ma and Yang 2008] and [Li et al. 2010].

distribution contention arises. Fig. 1 shows three data generators DG_1, DG_2, and DG_3 with overlapping offloading areas. The challenge is to resolve such contention while still achieving energy-efficient data redistribution.

Specifically, we formulate the data redistribution problem as a graph-theoretic problem. We show that the problem is APX-hard for arbitrary data sizes and that it is equivalent to minimum cost flow problem [Ahuja et al. 1993; Papadimitriou and Steiglitz 1982] for unit data sizes, which can be solved optimally. Due to the centralized nature of the optimal solution, we design a fully distributed algorithm for data redistribution that still achieves near-optimal performance. We model the sensor network as an electrostatic potential field, wherein the data generators correspond to electrical point charges, and 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 that perform comparable to the optimal solution.

The main results and contributions of this paper include the following:

(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 prove that the data redistribution problem is APX-hard for arbitrary data sizes and that it is equivalent to the classic minimum cost flow problem for unit data sizes.

(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 the TOSSIM simulator, we show that the distributed algorithm significantly outperforms EnviroStore [Luo et al. 2007], the only existing data redistribution scheme in sensor networks, in terms of both solution quality and message overhead.

(5) We extend the proposed distributed algorithm to take into consideration the individual energy consumption at each sensor node and to avoid disproportionate energy expenditure among sensor nodes, without compromising on the total redistribution cost in the network.

Paper Organization. The rest of the paper is organized as follows. Section 2 discusses the related work. In Section 3, we present the network model and energy model, formalize the data redistribution problem, and illustrate it with a simple example. We show that the problem is APX-hard for arbitrary data sizes. For unit data sizes, we show that the problem is equivalent to the minimum cost flow problem and discuss a few centralized optimal solutions and their complexities. Section 4 presents the potential field-based distributed algorithm and discusses a few centralized heuristics. In Section 5, we compare all the algorithms in terms of total redistribution cost and message overhead, and demonstrate the suitability of the proposed scheme under various network scenarios. In addition, we compare the proposed distributed algorithm with EnviroStore and also extend the algorithm to consider individual sensor energy consumption. Section 6 summarizes the results and discusses future research directions.

2. RELATED WORK

Luo et al. [Luo et al. 2007] present a cooperative storage system for sensor networks called EnviroStore, to improve the utilization of 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 proposed are heuristic-based and lack a rigorous performance analysis. 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 [Pinar and Hendrickson 2004; Siegel and Siegel 2008] and disk storage [Khuller and ah Kim 2003]. This problem 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. The data redistribution problem, however, is concerned with the storage space utilization as well as minimization of data redistribution energy in sensor networks.

The idea of potential functions has been adopted in sensor networks to address various issues (see [Toumpis 2008] for a good survey paper). The approach has been utilized to study how to route packets from source to destination in order to avoid congestion in anycast [Lenders et al. 2008] or multipath routing [Nguyen et al. 2004], or to study the placement of mobile sinks in wireless sensor networks for energy balancing [Toumpis 2008]. In all these problems, there are particular traffic sources and sinks. In the data redistribution problem, we have traffic sources (data generators) while the challenge is to find the appropriate sink nodes to redistribute the data from traffic sources. The goal is to efficiently utilize the storage capacity in order to reduce the redistribution energy cost, which is different from the above problems.

Recently Gao et al. [Gao et al. 2009] 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 within a logarithmic distortion factor. Internal nodes are used to match resources and events in a subtree. Unmatched resources or events are propagated up the tree until matched. Such preprocessing is not quite feasible in our case and also leads to a centralized solution approach. In addition, the required redistribution scheme should be resilient towards node failures, energy depletion, and storage depletion at the individual sensor node, which is the focus of our paper.

The data redistribution problem bears a resemblance to the graph Voronoi diagram problem [Erwig and Hagen 2000] in the sense of "areas of influence." The graph Voronoi diagram is the graph theory equivalent of the Voronoi diagram in

computational geometry. It characterizes regions of proximity in graphs based on shortest paths between nodes. Yet there are two differences between the data redistribution problem and the graph Voronoi diagram problem. First, in the data redistribution problem, each data generator node has a "weight," which indicates the amount of data to be redistributed. Second, the graph Voronoi diagram does not consider the "capacity" of each node, which, in our problem, signifies the available storage space of sensor nodes.

3. DATA REDISTRIBUTION PROBLEM

In this section, we elaborate on the network model, the energy consumption model and formulate the problem using a graph-theoretic framework. Then, we outline this problem's complexity and solution direction in the cases of arbitrary and unit data sizes respectively.

3.1 Network Model and Problem Formulation

In a data intensive sensor network (DISN), a subset of sensor nodes generate large amounts of sensory data over time. Each sensor node has limited storage capacity and can only hold a finite amount of sensory data items. Sensor nodes that collect more data than they can store in their local storage are the data generators, and they must redistribute/offload some of their data to other nodes that have available storage space. The sensory data are modeled as a sequence of raw data items, each of which has the same unit size, and the storage capacity of each node is an integral multiple of this unit size.[3] The objective of the 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.[4]

Network Model. Consider 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 that the sensor nodes are distributed uniformly at random in the deployment region since near-uniform node deployment is an easy and practical approach to providing full sensing coverage and connectivity, and is 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.

Let p denote the number of data generators in the network. Without loss of generality, we assume that the data generators are nodes $\{1, 2, ..., p\}$. The data generator i is referred to as DG i. Let s_i denote the number of data items DG i needs to redistribute, and let 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

[3]We also consider the scenario with arbitrary data sizes and show that the problem is APX-hard. However, for a typical sensor network application, the data could be divided into equally (unit) sized chunks that are much smaller than the sensor storage space. Therefore, during most of the discussion, we assume unit data size and that the storage capacity of each node is an integral multiple of this unit size.

[4]Later we also extend the proposed distributed algorithm to address individual node's energy concerns without compromising on the performance quality.

$m_i = 0$, meaning node i has a full storage space and thus cannot store any more data items; in this case, node i is a data generator. If $s_i = 0$, then $m_i \geq 0$, and node i can store m_i data items offloaded from other data generator nodes.

Energy Model. Each data generator redistributes one data item at a time. To model the energy cost, we use the number of hops to measure the energy consumption of redistributing the data item. We adopt the first order radio model [Heinzelman et al. 2000] wherein for k-bit data over distance l, the transmission energy $E_{Tx}(k,l) = E_{elec} \times k + \epsilon_{amp} \times k \times l^2$, and the receiving energy $E_{Rx}(k) = E_{elec} \times k$, where E_{elec} and ϵ_{amp} are constants. 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. [Liu and Cao 2010] and Nuggehalli et al. [Nuggehalli et al. 2003] 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.

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 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, i.e., $\sum_{i=1}^{p} s_i \leq \sum_{j=p+1}^{N} m_j$.

Problem Formulation. Let I denote the set of data items to be redistributed in the entire network, and let $S(i)$, where $i \in I$, denote the data generator of data item i. A *redistribution function* is defined as $r : I \rightarrow V$, indicating data item $i \in I$ is redistributed to node $r(i) \in V$ via a shortest path between $S(i)$ and $r(i)$. The objective is to find a redistribution function r that minimizes the total redistribution cost given by $\sum_{i \in I} d_{S(i)r(i)}$, i.e.,

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

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

$$|\{i | i \in I, r(i) = j\}| \leq m_j, \qquad \forall\ j \in V.$$

Let us consider an example to illustrate the data redistribution problem.

EXAMPLE 1. Fig. 2 illustrates the data redistribution problem in a small linear sensor network. Each edge is 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.

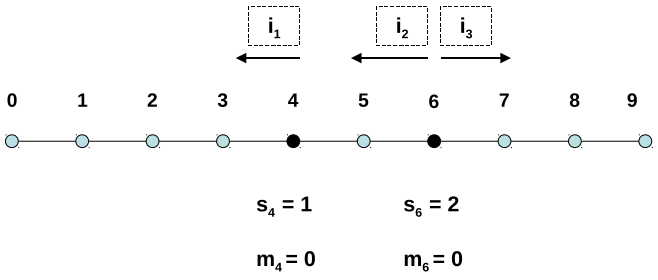

Fig. 2. Illustration of data redistribution problem in a linear network. All sensor nodes have unit storage capacity.

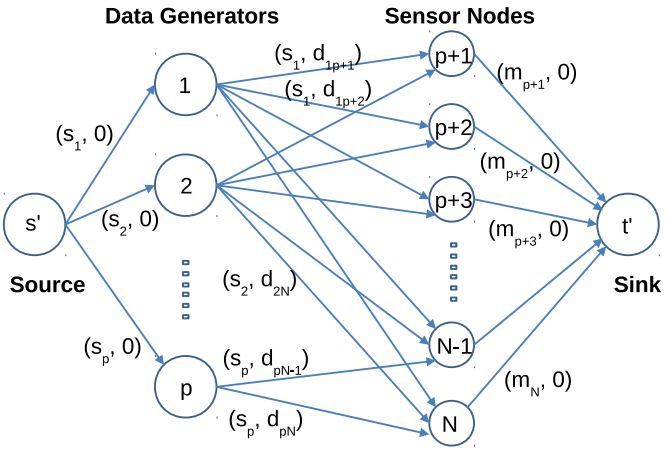

Fig. 3. Data redistribution problem with unit data sizes is equivalent to minimum cost flow problem.

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 minimum redistribution cost is 3. □

Next, we show that the redistribution problem with arbitrary data sizes is APX-hard; therefore, it is unlikely to find a polynomial time approximation algorithm.

THEOREM 1. *The data redistribution problem with arbitrary data sizes is APX-hard.*

Proof: We construct a special case of this problem and show that it is APX-hard. This is done by reducing the maximum 3-bounded 3-Dimensional matching (3DM-3) problem to this special case. 3DM-3 problem has been shown to be APX-hard [Kann 1991]. The detailed proof is provided in the Appendix. ∎

3.2 Minimum Cost Flow Problem

We show that the data redistribution problem with unit data sizes is equivalent to the minimum cost flow problem.[5] Recall that minimum cost flow problem [Ahuja et al. 1993; Papadimitriou and Steiglitz 1982] is the following. Consider 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 2. *The data redistribution problem with unit data sizes is equivalent to the minimum cost flow problem.*

Proof: Given the general sensor network graph $G(V, E)$, let $V_1 = \{1, 2, ..., p\}$ denote the set of p data generators, and V_2 denote the set of the rest $(N - p)$ sensor nodes, $V_1 \cup V_2 = V$. We transform 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):

1. Let $V' = V \cup \{s'\} \cup \{t'\}$, where s' is the new source node, and t' is the new sink node.
2. Let $E' = \{(i, j) : i \in V_1 \text{ and } j \in V_2\} \cup \{(s', i) : i \in V_1\} \cup \{(j, t') : j \in V_2\}$.
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 the 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' to $\sum_{i=1}^{p} s_i$. Set the supply and demand of other nodes in V' to 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 that $\sum_{i=1}^{p} s_i \leq \sum_{j=p+1}^{N} m_j$. Therefore, solving the minimum cost flow problem on $G'(V', E')$ provides the minimum redistribution cost solution for the data redistribution problem in $G(V, E)$.

■

The minimum cost flow problem can be solved efficiently in polynomial time using well-known algorithms [Goldberg 1997; 2008; Goldberg and Tarjan 1990; Ahuja et al. 1992; Orlin 1990; Tardos 1985; Hoppe and Tardos 2000]. In this paper, we use the algorithm and implementation by Goldberg [Goldberg 1997; 2008] 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, $C = \max\{\max_i s_i, \max_j m_j\}$. If C is not very large, Goldberg's algorithm is suitable to solve the minimum cost flow problem. Otherwise, other strong polynomial algorithms such as [Orlin 1990] $(O((MlogN)(M + NlogN)))$ and [Tardos 1985] $(O(M^4))$ can be used.

However, the optimal solution's centralized nature makes it unsuitable for large-scale distributed sensor network deployment. Therefore, in the next section, we

[5]Note that such equivalence is not possible in the case of arbitrary data sizes.

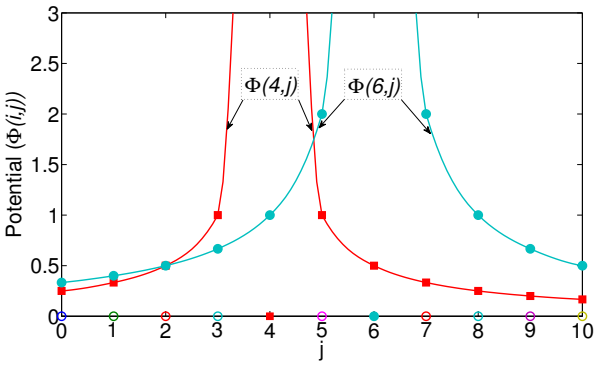

Fig. 4. Potential field of the sensor network in Example1.

design a distributed algorithm to address this problem. The proposed algorithm relies on potential field based computations to determine the redistribution function r in a distributed manner.

4. POTENTIAL-BASED DISTRIBUTED ALGORITHM (PDA)

Let us revisit 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 to node 7. However, in a distributed environment, since both node 3 and node 5 are the same distance to node 4, node 4 could offload its data to node 5, resulting in a 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 the proposed potential-based distributed algorithm (also referred to as PDA), followed by the discussion of its convergence and performance.

4.1 Potential Field Model

We study data redistribution using an analogy, modeling the entire sensor network as an electric potential field wherein each data generator is an electric charge. DG i with s_i data items to offload has a positive electric charge of s_i. For arbitrary sensor node j, its potential due to DG i, denoted as $\phi(i,j)$, equals $k_0 \frac{s_i}{d_{ij}}$, where k_0 is a constant, and d_{ij} is the distance between DG i and node j (we omit k_0 in the rest of the paper since it is a constant and cancels out in all comparisons). According to the superposition principle [Jackson 1999], 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, denoting its total potential as $\phi(j)$, we obtain:

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

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 the 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, i.e., $\phi(6,1)\left(= \frac{2}{5}\right) > \phi(4,1)\left(= \frac{1}{3}\right)$, 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 the 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 decide to offload its data item to node 3 due to its lower total potential.

4.2 Potential-Based Distributed Algorithm

The potential-based distributed algorithm (PDA) takes place in iterations. Each iteration consists of the following three stages:

1. **Advertisement Stage.** Each data generator DG i that has data items to redistribute floods an advertisement message in the network containing its ID and number of data items to offload (s_i).[6] An integer (initialized as 0) is also 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. Every node forwards the advertisement message if it is the first time it receives it.
2. **Storage Commitment Stage.** Let c_{kj} denote the amount of storage space node j has committed to DG k. Initially, $c_{kj} = 0, 1 \leq k \leq p$. Each sensor node j with available storage space $m_j > 0$, after receiving advertisement messages from all the DGs, performs the following steps:
   - A. Computes its potential value due to DG i as $\phi(i,j) = \frac{s_i}{d_{ij}}$. $\phi(i,j) = 0$ if j did not receive DG i's advertisement message. It also computes its total potential $\phi(j)$ after receiving all advertisement messages.
   - B1. Finds the data generator that corresponds to the maximum potential value. Ties are broken randomly. Suppose that data generator is DG k where $k = \text{argmax}_{1 \leq i \leq p} \phi(i,j)$. Then node j *commits* one unit of storage space to DG k and updates $m_j = m_j - 1$, and $c_{kj} = c_{kj} + 1$.
   - B2. If $m_j > 0$, j still has free storage space to commit, so 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 total potential $\phi(j)$, and d_{ij}. Note d_{ij} has been obtained in Stage 1.

[6] 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 $O(N)$.

3. **Data Offloading Stage.** In this stage, each data generator decides the number of data items to offload to each committed sensor node. We denote the set of sensor nodes who commit storage to DG i as $\mathcal{C}_i$. After receiving all commitment messages from nodes in $\mathcal{C}_i$, each DG i performs the following computations:
   A. Compares the total number of received commitments, $\sum_{j\in\mathcal{C}_i} c_{ij}$, 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 to handle the scenario where more commitments are received at the data generator than required. Otherwise, DG i can completely satisfy all the commitments and 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. Decides how many data items to offload and 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 committed sensors in $\mathcal{C}_i$. If there are multiple closest nodes, it breaks 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 $c_{ik} = 0$ (DG i has decided to offload as much data to node k as committed by node k during storage commitment stage), then it removes k from set $\mathcal{C}_i$.
   B2. If DG i still has data to redistribute ($s_i > 0$), then 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 data items to sensor nodes according to the above calculation.

4.3 Discussion of PDA

The PDA stops when all the data generators have offloaded their data items. In each iteration, each sensor commits all its storage space. 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 sensor nodes no longer having storage space available and data generators no longer having data items to offload do not actively participate in the next iteration, other than forwarding messages.

It is easy to check that the PDA solves Example 1 optimally in just one iteration. The PDA takes place in iterations; thus, some synchronization is needed, which could be achieved by associating a time interval with each stage (and with each iteration). Essentially, the PDA is a fully distributed, highly scalable, and efficient data redistribution mechanism with the following characteristics:

—The PDA is an online distributed algorithm and is applicable in environments where data generation occurs dynamically. It does not require 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 the 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 generators' redistribution needs.

—The PDA is adaptable to network dynamics such as dynamic data generation, node failures, energy and storage depletion. For instance, the sensor nodes could change roles from DG to non-DG or vice-versa, the data generation rate at a DG could change over time, etc. We refer to such scenarios as *dynamic data generation* scenarios. Our proposed schemes are applicable in the dynamic data generation scenarios as long as the scenario remains same for some period of time before changing again. For instance, using PDA, if the change occurs after an iteration is over, the next iteration of the PDA would be able to incorporate the changed scenario.

Note that in the storage commitment stage, each sensor node commits all its free storage space, even when the total number of data items advertised in the received advertisement messages is less than the size of its storage space. 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 the PDA.

The PDA is more applicable in challenging scenarios where the number of data items and number of data generators are large. In sparse networks (where the number of data generators and/or the number of data items are few), the PDA could be modified to reduce the complexity by sending out advertisements to a few hops only. For example, a time-to-live (TTL) value can be included in the message indicating how many hops the message has traveled. It allows the flooding scope to be limited by only rebroadcasting messages that have a TTL value greater than zero.

Routing Support in PDA. In both storage commitment and data offloading stages, nodes rely on some routing information to forward messages. Such information can be readily obtained in advertisement and commitment stages without costing additional overhead messages. In the advertisement stage, when a node receives the advertisement message of a data generator from a neighbor, it records the neighbor as the next hop to the data generator as well as the distance to the data generator. This information is updated if another advertisement message from the same data generator with shorter distance arrives. In this way, a "routing table" with entries to each of the data generators is maintained at each node.[7] This routing information is used in the storage commitment stage to route the commitment messages back to the data generators using the shortest path between the node and data generator. In the storage commitment stage, when a node receives commitment messages from a neighbor, it also records the neighbor as the next hop to the committing sensor node, and this information is used for data delivery in the data offloading stage.

4.4 Performance and Convergence Analysis of PDA

Next, we analyze the convergence and performance of the PDA, followed by its message and time complexity analysis. Note that this analysis does not consider message losses and delays in detail. A detailed analysis of PDA in the presence of

[7]Note that the routing table here is not a full-fledged one; each node only maintains routing entries for each DG.

losses and delays is an interesting topic of future research.

4.4.1 *Convergence Analysis*

THEOREM 3. *PDA stops in at most p iterations, where p is the number of data generators in the network.*

Proof: We first show by contradiction that in each iteration, at least one data generator receives more commitments than the number of data items it needs to redistribute. Assume that in the first iteration of the PDA, none of the data generators receive more commitment than number of its data items. That is,

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

Summing over all 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 the assumption that the total size of the data items to be redistributed by 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, the PDA takes at most p iterations.

■

4.4.2 *Performance Analysis.* Let c_i denote the total number of commitments received by DG i during an arbitrary iteration of PDA. That is, $c_i = \sum_{j \in \mathcal{C}_i} c_{ij}$.

THEOREM 4. *If $m_j \in \{0, 1\}$, $\forall j \in \{p+1, \ldots, N\}$, $s_i = s$, $\forall i \in \{1, \ldots, p\}$, where s is a constant such that $p \cdot s \leq \sum_{j=p+1}^{N} m_j$, and $c_i \leq s_i$, $\forall i \in \{1, \ldots, p\}$, during an arbitrary iteration of PDA, then $\sum_{i=1}^{p} c_i$ data items get redistributed optimally during this iteration.*

Proof: If $c_i \leq s_i$, $\forall i \in \{1, \ldots, p\}$, then all the commitments are utilized using the PDA, and each DG i redistributes c_i data items during this iteration. In total,

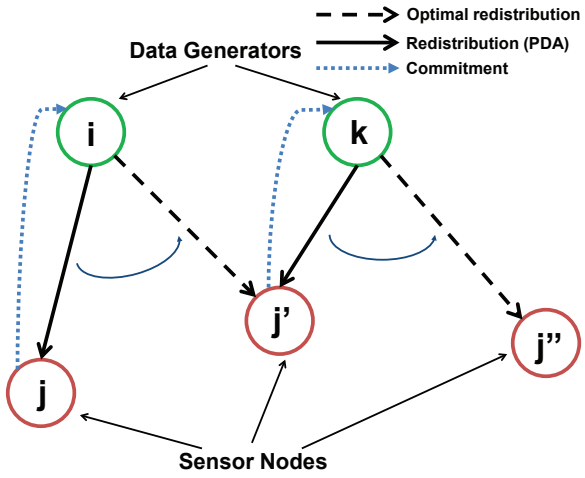

Fig. 5. Sensor nodes j and j' commit storage space of one data item each to data generators i and k, respectively. Redistribution using PDA involves DG i redistributing one data item to node j and DG k redistributing one data item to node j'. Optimal redistribution, however, involves DG i redistributing one data item to node j' and DG k redistributing one data item to node j''.

$X = \sum_{i=1}^{p} c_i$ data items are redistributed. We show using contradiction that all these data items get redistributed optimally.

Let us assume that the data redistribution achieved above using the PDA is non-optimal. This implies that it is possible to redistribute X data items at a strictly lower redistribution cost compared to that incurred by the PDA. Thus, there exists at least one data item (out of X data items), whose redistribution hop count could be reduced further. Let this data item belong to DG i, for some $i \in \{1, \ldots, p\}$. Let the sensor node to whom this data item is getting redistributed in the PDA be denoted as j, for some $j \in \{p+1, \ldots, N\}$. Since the redistribution of one data item from DG i to sensor node j is non-optimal, under optimal redistribution, DG i does not redistribute one data item to node j, and redistributes one data item to some node j' ($j' \in \{p+1, \ldots, N\}, j' \neq j$) instead. This implies,

$$d_{ij} > d_{ij'}. \tag{3}$$

Figure 5 depicts the scenario in detail. Note that node j' must have available storage space, and due to the assumption that $m'_j \in \{0, 1\}$, $m_{j'} = 1$. Thus, node j' commits storage for one data item to exactly one data generator during the storage commitment phase. However, node j' did not commit to DG i (or else DG i would be redistributing one data item to node j' as well), and committed its storage space to some other data generator k ($k \in \{1, \ldots, p\}, k \neq i$) instead. This implies,

$$\frac{s_k}{d_{kj'}} \geq \frac{s_i}{d_{ij'}}. \tag{4}$$

Since, according to PDA, DG i redistributes one data item to node j, node j committed its storage space of one data item to DG i (and not to DG k). This

implies,

$$\frac{s_i}{d_{ij}} \geq \frac{s_k}{d_{kj}}. \tag{5}$$

Using equations (3), (4), and (5), and the fact that $s_i = s_k = s$, we have,

$$d_{kj'} \ \leq \ d_{ij'} \ < \ d_{ij} \ \leq \ d_{kj}. \tag{6}$$

Now, since $c_k \leq s_k$ and $m_{j'} = 1$, if DG i redistributes one data item to node j' (instead of redistributing to node j) under optimal redistribution, then DG k would need to redistribute one data item to some other node j'' ($j'' \in \{p+1, \ldots, N\}, j'' \neq j'$).

Case I: $j'' = j$

If node j'' is the same as node j, then the cost savings achieved by this optimal redistribution, compared to the original redistribution using PDA, is given by,

$$d_{ij} - d_{ij'} + d_{kj'} - d_{kj} = (d_{ij} - d_{ij'}) - (d_{kj} - d_{kj'}) \leq 0. \tag{7}$$

The inequality above follows from (6). Thus, the optimal redistribution cost is at least as large as the redistribution cost using PDA, and we reach a contradiction.

Case II: $j'' \neq j$

If node j'' is different from node j, the argument in Case I above is extended to show a contradiction. Note that node j'' must have available storage space, i.e., $m_{j''} = 1$. Thus, node j'' commits storage for one data item to exactly one data generator during the storage commitment phase. However, node j'' did not commit to DG k (or else DG k would be redistributing one data item to node j'' as well), and committed its storage space to some other data generator l ($l \in \{1, \ldots, p\}, l \neq k$) instead. (Note that $l = i$ is possible, and the analysis applies to this case as well.) This implies,

$$\frac{s_l}{d_{lj''}} \geq \frac{s_k}{d_{kj''}}, \ \text{ or } \ d_{kj''} \geq d_{lj''}. \tag{8}$$

If $l \neq i$, since node j committed to DG i and not to DG l, we have,

$$\frac{s_i}{d_{ij}} \geq \frac{s_l}{d_{lj}}, \ \text{ or } \ d_{lj} \geq d_{ij}. \tag{9}$$

Both (8) and (9) hold if $l = i$. Now, since $c_l \leq s_l$ and $m_{j''} = 1$, if DG k redistributes one data item to node j'' (instead of redistributing to node j') under optimal redistribution, then DG l would need to redistribute one data item to some other node j''' ($j''' \in \{p+1, \ldots, N\}, j''' \neq j''$). Now, if node j''' is the same as node j ($j''' = j$), then the cost savings achieved by this optimal redistribution, compared to the original redistribution using the PDA, is given by,

$$d_{ij} - d_{ij'} + d_{kj'} - d_{kj''} + d_{lj''} - d_{lj} = (d_{ij} - d_{lj}) - (d_{ij'} - d_{kj'}) - (d_{kj''} - d_{lj''}) \leq 0. \tag{10}$$

The inequality above follows using (6), (8), and (9). Thus, the optimal redistribution cost is at least as large as the redistribution cost using the PDA, and we reach a contradiction.

If $j''' \neq j$, then the argument can similarly be extended to show a contradiction. Since, the number of data generators and sensor nodes is finite, and X amount of data items can be redistributed, some data generator must redistribute to node j under the optimal redistribution scheme. ∎

Note that we had earlier assumed that $\sum_{i=1}^{p} s_i \leq \sum_{j=p+1}^{N} m_j$. Thus, the condition $c_i < s_i,\ \forall i$ could only occur if the network gets disconnected or upon node failures. However, $c_i = s_i,\ \forall i$ could also occur when the storage space in the network is limited or is running out. In this case, all the data items get redistributed optimally, from Theorem 4. This leads us to the following result:

COROLLARY 1. *If* $m_j \in \{0,1\},\ \forall j \in \{p+1,\ldots,N\},\ s_i = s,\ \forall i \in \{1,\ldots,p\}$, *where* s *is a constant such that* $p \cdot s \leq \sum_{j=p+1}^{N} m_j$, *and* $c_i = s_i,\ \forall i \in \{1,\ldots,p\}$, *during an arbitrary iteration of the PDA, then the PDA achieves optimal performance, i.e. all data items get redistributed optimally during this iteration.*

Thus, the PDA performs optimally under extreme conditions such as disconnected network, sensor node failures, and lack of abundant storage space in the network. Under other scenarios, the PDA achieves close to optimal performance, as we show in the next section.

4.4.3 *Message and Time Complexity Analysis.* Next, we analyze the message and time complexity of the PDA.

Message Complexity. We compute the message complexity of the PDA by calculating the total number of transmissions incurred in the PDA. During each iteration there are at most p data generators broadcasting advertisement messages, reaching all N nodes. There are at most p iterations from Theorem 3. 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 $\mathrm{O}(p^2N^2)$. During the data offloading stage, there are total $\sum_{i=1}^{p} s_i \leq N\overline{m}$ data items to offload, each travels at most N hops, resulting in total $O(N^2\overline{m})$ offloading transmissions, where $\overline{m}$ is the average storage capacity of each sensor node. So the total number of transmissions (or message overhead) in PDA[8] is $O((p^2 + \overline{m})N^2)$.

Next we compare the message overhead of the PDA with the message overhead of a centralized approach. In this approach, a centralized coordinator, which can be assumed to be the dedicated sensor in the network, collects all the global information, computes the solution in a centralized manner(by solving the corresponding minimum cost flow problem), and announces the relevant part of the solution back to each sensor node. First, every data generator sends a message to the coordinator, informing the number of data items it has to offload; and every non-data generator node sends a message to the coordinator, informing its storage capacity. The total number of such messages is of the order $O(N^2)$. Second, after receiving all the information, the coordinator performs a minimum cost flow computation, and finds out the redistribution function to optimally offload each data item from its data

[8]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 5.

Table I. Time complexity comparison of all data redistribution algorithms.

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

generator into the network. It then informs each data generator its redistribution strategy, which takes $O(pN)$ messages. Third, after receiving such information from the coordinator, each data generator offloads its data to other nodes following the instruction from the coordinator, resulting in total $O(N^2\overline{m})$ offloading transmission messages. Therefore, the total number of messages in this centralized approach is $O(N^2\overline{m})$. If $p^2 < \overline{m}$, the message complexity of PDA is better than that of the centralized approach. When the number of data generators is large, the message overhead of the PDA is larger than that of this centralized approach. However, the proposed PDA is more desirable since the centralized coordinator is a central point-of-failure. Its failure affects the computation of the data redistribution strategy and compromises the entire system. Furthermore, selecting a new coordinator and informing the whole network about it also involves flooding and costs many communication messages, which could result in larger message overhead than the PDA.

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

4.5 Centralized Heuristics

We design a set of centralized and intuitive heuristics, namely the random algorithm (Random), greedy algorithm (Greedy), and cooperative algorithm (Cooperative), and later compare their performances with the PDA and the Optimal. The Optimal solution is obtained by solving the minimum cost flow problem using the algorithm and implementation by Goldberg [Goldberg 1997; 2008].

In the Random, each data generator randomly selects a sensor node with available storage space to offload one of its data items. This process is repeated for all its data items. Its time complexity is $O(\sum_{i=1}^{p} s_i N) = O(\bar{m}N^2)$. This is the most efficient heuristic in terms of time complexity.

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

The Cooperative takes place in rounds. In each round, similar to the Greedy, the

data generators take turns 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 the Greedy, i.e., $O((p + \bar{m})N^2)$; however, this heuristic tends to perform much better than the Greedy, as we show in Section 5.

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 the Optimal) are not suitable for distributed environments, and are considered mainly for performance comparisons with PDA.

5. PERFORMANCE EVALUATION

In this section, we first compare the performance of the PDA and other centralized heuristics with that of the optimal solution for the minimum cost flow problem. We then compare the PDA with EnviroStore, an existing data redistribution scheme in sensor networks. Finally, we extend the PDA to avoid disproportionate energy consumption at individual sensor nodes, without compromising the solution quality.

5.1 Comparison among Optimal, PDA, and Other Heuristics

We first visually compare the performances of different algorithms. For this purpose, we assume that each sensor node has one unit storage space, and we adopt a sensor network with grid-like topology (note that our proposed algorithm is 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 (with up to 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. 6-10 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), and (12, 9), respectively. 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, as computed by the algorithm used. We observe that under the Greedy, the data generators took turns to redistribute their data items in the following order: circle, square, diamond, followed by triangle. We observe that both Cooperative and PDA perform close to Optimal. However, the performance of the 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 the PDA and Cooperative are able to successfully estimate this inherent structure. Thus, the Cooperative outperforms the Greedy by forcing DGs to redistribute one data at a time. The difference between the structure of offloading area in the Optimal solution and when using the PDA (and the Cooperative) is that the sensor nodes with x-coordinate of 10 are equidistant

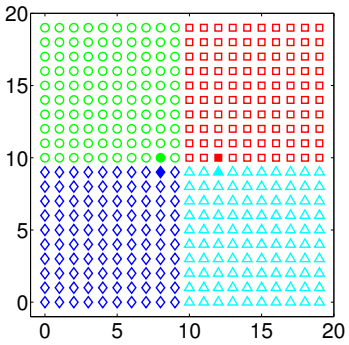

Fig. 6. Optimal (redistribution cost = 3,160).

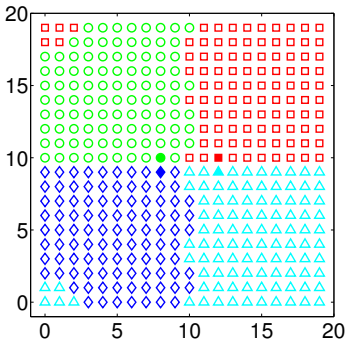

Fig. 7. PDA (redistribution cost = 3,205).

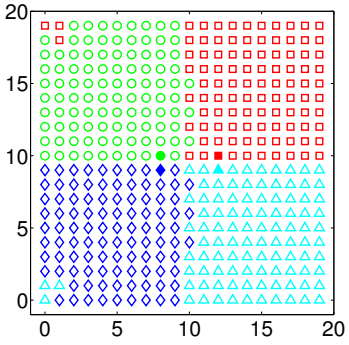

Fig. 8. Cooperative (redistribution cost = 3,200).

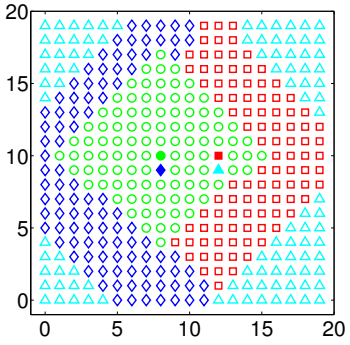

Fig. 9. Greedy (redistribution cost = 3,524).

(in terms of hop count of the shortest distance) from exactly two data generators. In the PDA, this causes these sensor nodes to commit randomly to one of the two data generators (since $s_i = 99$ for all four DGs, the resulting potential at these sensor nodes is the same for two DGs). This behavior is the key reason why the PDA performance is slightly worse than that of Optimal. Note that the difference in performance between the PDA and the Cooperative is not large in this case. However, as we shall show later in Fig. 13, when the size of the problem is sufficiently large, the PDA outperforms the Cooperative. In addition, note that the 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; and 3) all data generators are randomly placed in the network. Fig. 11 compares

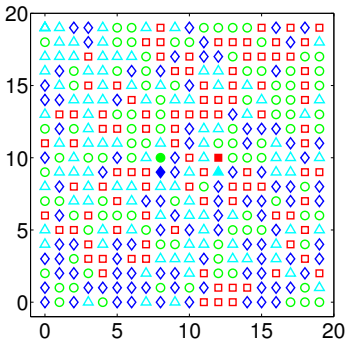

Fig. 10. Random (cost = 5,235).

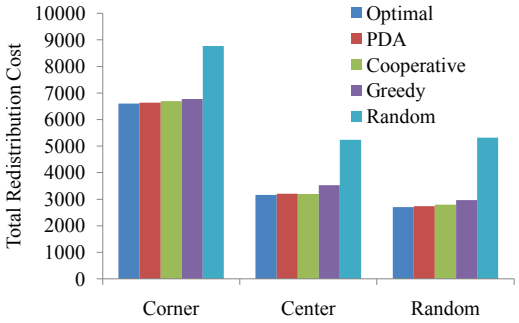

Fig. 11. Performance with different DG locations.

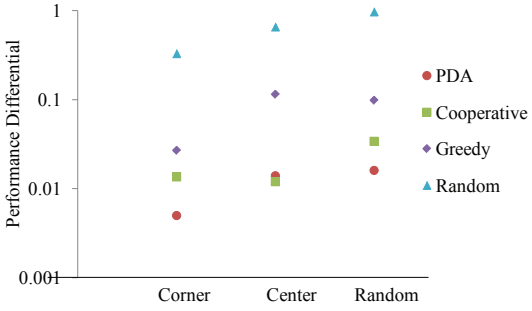

Fig. 12. Performance percentage differential under different scenarios.

the total redistribution cost of each algorithm under these scenarios. We observe that all algorithms incur the largest redistribution cost when all data generators are at one corner, followed by when all 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 must 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 the 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 the Optimal, divided by the cost of the Optimal. Fig. 12 shows the PPD of the Greedy, Random, Cooperative, and PDA. We observe that the performance difference of the PDA is within 5% of the optimal performance in all the scenarios. Also note that as the location of data

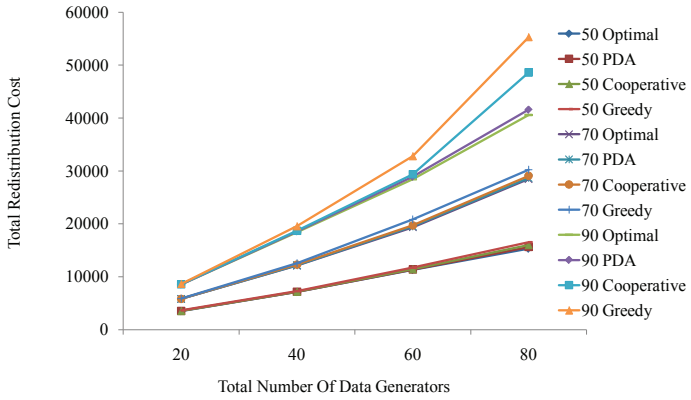

Fig. 13. Varying number of data generators and data items in 100×100 network. The legend '50 Optimal' corresponds to the redistribution cost incurred by Optimal when each data generator has 50 data items to redistribute.

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 the PDA or the heuristics.[9]

Varying Number of Data Generators and Number of Data Items. Next, we study the performance comparison by varying number of data generators and data items to be redistributed. We consider a 100×100 network, with 10,000 sensor nodes, each with unit storage capacity. We vary the number of data generators from 20, 40, 60, and 80 and the total number of data items of each data generator from 50, 70, and 90. The data generators are randomly placed in the network. From Fig. 13, we observe that the PDA performs comparable to the Optimal, and the Greedy performs worse than the Cooperative. Note that the redistribution cost of the Random is very high in all cases, and is hence omitted. The difference between the redistribution algorithms is seen clearly when there are more data generators with a 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 the PDA. Therefore, the PDA is suitable for large scale data-intensive sensor networks, with heavy data redistribution requirements.

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

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

5.2 Comparison between PDA and EnviroStore

We implement the PDA using TinyOS 2.1 and compare it with EnviroStore using the TOSSIM simulator.[10] 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 a 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 in advertisement messages (e.g., once per minute). When the remaining storage size incurs big change (of more than the node advertisement threshold) since the last advertisement, the node sends an extra advertisement message immediately to ensure accurate and timely knowledge of such information by its neighbors. When its remaining storage is less than a particular threshold and the imbalance (i.e., 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 storage-underloaded neighbors. However, if several nodes simultaneously offload data to the same node, then 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 a reliable unicast for nodes to transfer data. The simulations in [Luo et al. 2007] show significant improvement in the amount of data collected in the network using EnviroStore compared with the one without any redistribution mechanism.

Differences between PDA and EnviroStore. There are several differences between PDA and EnviroStore. First, EnviroStore is broadcast-based, which requires constant message exchanges among neighboring nodes. This incurs a large amount of message overhead. In PDA, broadcasting is only adopted in advertisement stage. Second, no routing information needs to be maintained at nodes in EnviroStore as an advantage of broadcast, while some necessary routing information needs to be maintained in PDA to route commitment messages to data generators and to route data offloading messages to sensor nodes. However, as noted in Section 4.3, such information can be easily obtained in the stages of advertisement and storage commitment and thus no additional messages are needed in PDA to set up the routing information. Third, EnvitoStore is asynchronous, whereas in PDA, certain synchronization is needed for the different stages and iterations to be performed correctly.

Simulation Setup. We adopt the grid-like sensor deployment in [Luo et al. 2007], with 36 nodes placed in a grid manner with the only difference being that all 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

[10]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 performed here.

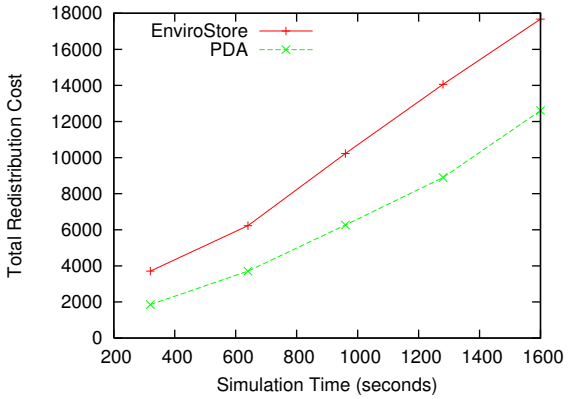

Fig. 14. Comparison of total redistribution cost between EnviroStore and PDA. The data generating rate is set to 64 bytes/sec in both schemes. The duration for each iteration in PDA is set to 80 seconds.

neighbors only. Among the 36 nodes, two nodes are configured as DGs and are located at (2, 2) and (5, 5), respectively, as shown in Fig. 15. The DGs periodically generate data and create input for both EnviroStore and PDA. For EnviroStore simulation, we adopt all default threshold values in [Luo et al. 2007], i.e., the remaining storage threshold is 0.95S (S is the total storage of a node, which is 16KB), imbalance threshold (between the average remaining storage of its neighbors and its own remaining storage) is 0.05S, and node advertisement threshold is 0.01S. The size of each data packet is set to 22 bytes in both PDA and EnviroStore simulations.[11] The data generating rate is set to 64 bytes/sec in both schemes. Like EnviroStore, reliable communication is implemented in PDA, in both commitment and data offloading stages.

Comparison of Redistribution Cost. First, we compare the total redistribution cost (using Equation 1) of EnviroStore and PDA at various values of simulation running time. The duration for each iteration in PDA is set to 80 seconds. Fig. 14 shows that the total redistribution cost of PDA is 30% to 50% smaller than that of EnviroStore for most values of the simulation time, indicating that PDA is more energy-efficient than EnviroStore. 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. 15 depicts this performance comparison in detail. Second, EnviroStore only requires each node to communi-

[11] In [Luo et al. 2007], 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 a 7-byte header).

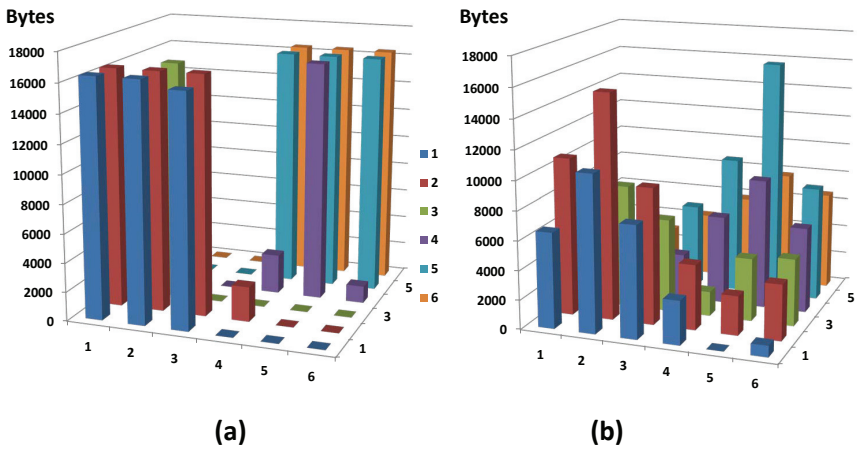

Fig. 15. 3D comparison of data redistribution dynamics when using (a) PDA, and (b) EnviroStore. Two DGs are located at (2, 2) and (5, 5), respectively. Here, the z-axis represents the amount of data redistributed to sensor nodes at different grid locations at simulation time = 3600 seconds. When using PDA, most of the data gets redistributed to nodes near the DG locations, unlike in EnviroStore.

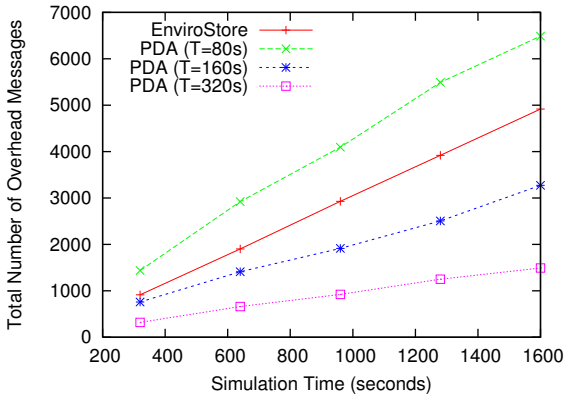

Fig. 16. Comparison between PDA and EnviroStore in terms of total number of overhead messages. The data generating rate is set as 64 bytes/sec. The duration for each iteration in PDA varies as 320, 160, and 80 seconds.

cate with its one-hop neighbors, whereas in PDA, the advertisement and storage commitment phases enable the effective communication between data generators and all other sensor nodes, leading to a more energy-efficient data redistribution. As a result, PDA is 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. The above comparison focused solely on the so-

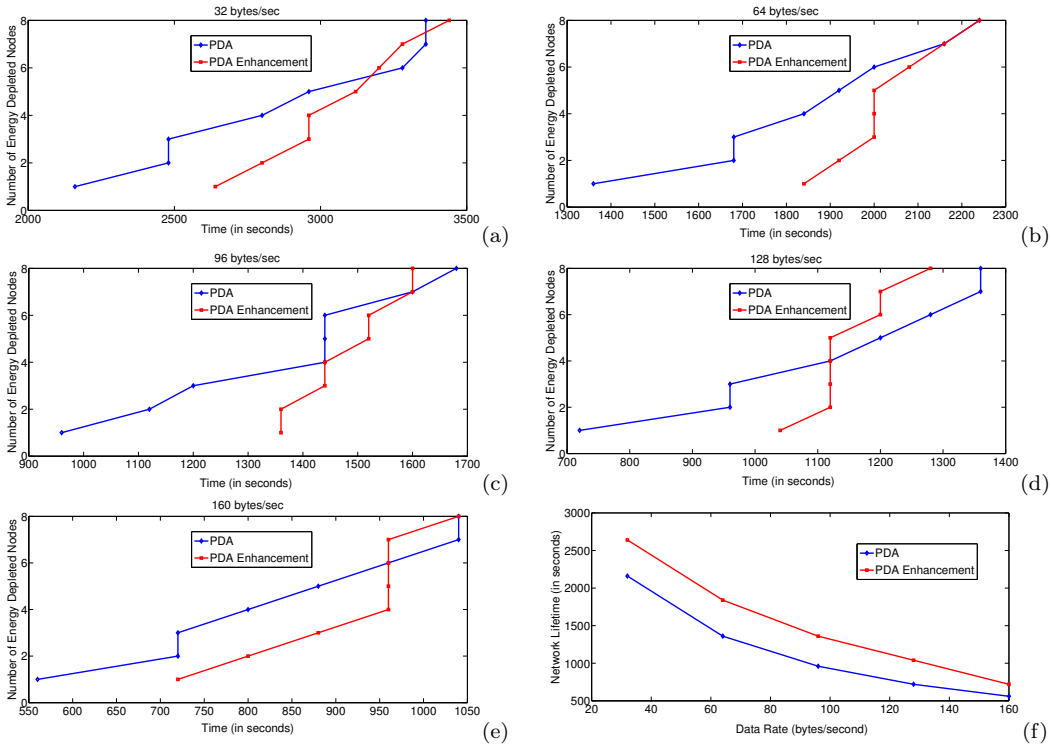

Fig. 17. Comparison of number of energy-depleted nodes over time under varying data generating rates: (a) 32 bytes/s, (b) 64 bytes/s, (c) 96 bytes/s, (d) 128 bytes/s, (e) 160 bytes/s, (f) network lifetime.

lution quality achieved using PDA and EnviroStore. Next, we discuss the message overhead incurred by the two schemes. The overhead messages in PDA include the advertisement messages and the storage commitment messages, while the overhead messages in EnviroStore include periodic advertisement messages sharing remaining storages among nodes. In the comparison, the data generating rate is set to 64 bytes/sec in both schemes. The duration for each iteration in PDA varies as 320, 160, and 80 seconds. The next iteration of PDA starts immediately after the previous iteration ends. From Fig. 16, we observe that the message overhead of PDA is less than that of EnviroStore in most scenarios. The reason is that PDA does not broadcast periodic advertisement messages. Instead, PDA incurs advertisement overhead at the beginning of each iteration. Therefore, a decrease in the time period of one iteration leads to an increase in the occurrence frequency of advertisement stages, resulting in larger overall message overhead. If the time period of one iteration is reduced further to 80 seconds, then the overhead incurred by PDA exceeds that of EnviroStore. Thus, PDA should be employed less frequently in order to reduce the message overhead. Note that in this comparison, we have not included the overhead caused due to the ping-pong phenomena in EnviroStore, which would result in additional message overhead for EnviroStore.

5.3 PDA Enhancement to Balance Individual Node Energy Consumption

In the proposed PDA, it is possible that some sensor nodes could get drained of their energy well before other nodes, causing a disproportionate energy consumption in the network. We enhance PDA by proposing mechanisms to avoid this without compromising the total redistribution cost. These mechanisms are built upon the routing support of PDA (described in Section 4.3) as follows. In the advertisement stage, when a node forwards the advertisement message, it adds its remaining energy level into the message. When a node maintains the shortest path information towards each DG, it also includes the remaining energy level of the next hop neighbor to the DG. Meanwhile, if there are multiple shortest paths to each DG, each sensor node stores all of them by recording each of the next hop neighbors to DG and their remaining energy levels. In the storage commitment stage, each sensor node sends its commitment message using the next hop neighbor with the maximum energy level, as do the intermediate nodes that relay the message. Note that a choice of next hop is considered only when there are multiple shortest paths from a node to a DG. Thus, the proposed mechanism does not change the redistribution energy cost in the network. A similar mechanism is followed to route the data offloading message from a DG to a sensor node.

Simulation Setup. The network topology is the same as in Section 5.2, with two data generators located at (2, 2) and (5, 5), respectively in a sensor network grid of size 6×6. The storage capacity of each sensor node is 16KB. The initial energy level of each node is randomly set between 1,000 and 2,000 units.[12] Each node costs 0.5 unit of energy when sending or receiving either a data or overhead message. This is consistent with our energy model where transmitting one packet (including sending and receiving) over one hop consumes one unit of energy. The iteration period is set to 80 seconds. The simulation stops when the network gets disconnected or the storage limit of the network is reached, since the DGs can no longer offload the data.

Simulation Results. We vary data generating rates of the two DGs as 32, 64, 96, 128, and 160 bytes/s and compare the number of energy-depleted sensor nodes for the PDA with and without the energy balance enhancement, as shown in Fig 17 (a) - (e). We note the following observations. First, for the PDA either with or without enhancement, with the increase of the data generating rate, the pace at which nodes deplete their energy increases. Specifically, to deplete the first eight nodes, for the PDA without enhancement, it takes around 3,400 seconds for 32 bytes/s, and around 1,040 seconds for 160 bytes/s; for the PDA with enhancement, it takes around 3,450 seconds for 32 bytes/s and around 1,040 seconds for 160 bytes. This is as expected, since a larger data generating rate incurs faster energy consumption in the network. Second, we observe that, in most cases, the number of energy depleted sensor nodes at a particular point of time in the network using the PDA is larger than that using the PDA with enhancement, indicating that the

[12]However, throughout the simulations, we notice that the two DGs deplete their energy much faster than other sensor nodes because they are much more heavily involved in message transmission during the redistribution process, and the simulation stops shortly after it begins. Therefore, to obtain more interesting results, the initial energy of the two DGs is set to infinity. We believe this does not affect the fair comparison of the two schemes (PDA and PDA enhancement).

PDA with enhancement successfully alleviates the energy consumption imbalance among sensor nodes, thus delaying the early energy depletion of some sensor nodes. Specifically, it shows that the energy depletion of sensor nodes in the PDA with enhancement is triggered later during the simulation. However, once one sensor node runs out of energy, the next node follows soon thereafter, and so on, resulting in a steep rise in the figures. Third, if we define the time when the first sensor depletes its energy as the *network lifetime* and compare these two schemes, as shown in Fig 17 (f), we observe that the lifetime achieved using PDA enhancement is larger than that achieved using the PDA at all values of data generating rate in the network. Therefore, the proposed PDA enhancement is able to achieve more uniform energy consumption rates at sensor nodes in the network, without compromising the solution quality in terms of redistribution cost.

6. 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 with unit data sizes is equivalent to the minimum cost flow problem, which can be solved optimally in a centralized manner. We also show that the problem is APX-hard for arbitrary data sizes. Second, we apply the notion of an electrostatic potential field to design a distributed data redistribution algorithm. We analyze the convergence and performance of the proposed algorithm and highlight its message and time complexities. Through simulations, we show that the distributed algorithm performs very close to the optimal centralized solution. In all scenarios considered, the difference between the optimal performance and the performance obtained by the PDA is less than 5%. Using the TOSSIM simulator, we show that the proposed distributed algorithm performs better than the existing data redistribution technique (EnviroStore), in terms of both the solution quality and message overhead. The PDA is extended to take the individual energy consumption of sensor nodes into account, and we show by simulation that it can successfully alleviate the disproportionate energy consumption among sensors.

We have only considered data redistribution due to storage overflow. Data redistribution would also be needed in case of energy depletion at the sensor nodes. Energy-depletion triggered redistribution would lead to multiple redistribution instances for the same data item over time, and the network lifetime would be an appropriate metric to optimize in that case.

We did not consider message losses or delays while analyzing and evaluating the proposed PDA. Since PDA requires some synchronization, large message delays could potentially result in overlapping iterations. Similarly, message losses could lead to slower convergence. These caveats are currently not considered and would be worthwhile to study both theoretically and experimentally in future.

7. ACKNOWLEDGEMENTS

This research was supported in part by NSF under grants CNS-1116849 and EPS-0903806. We also thank Drs. Joseph S.B. Mitchell and Esther M. Arkin, and the anonymous reviewers of TOSN and IEEE SECON/MASS 2010 for their very helpful suggestions.

APPENDIX – Proof of Theorem 1

Proof: To prove the APX-hardness of the data redistribution problem with arbitrary data sizes (DRPA), we construct a highly restricted special case of DRPA and show that it is APX-hard. This is done by reducing the maximum 3-bounded 3-Dimensional matching (3DM-3) problem to this special case of DRPA. The 3DM-3 problem has been shown to be APX-hard [Kann 1991].

3DM-3 problem. Consider three disjoint and unordered sets X, Y, and Z, where $|X| = |Y| = |Z| = n$, and a relation $T \subseteq X \times Y \times Z$. We denote by m the number of elements (hyperedges) in the set T, $|T| = m$. A matching in T is a subrelation (or subset) $M \subseteq T$ such that for all pairs of (x_i, y_i, z_i) and $(x_j, y_j, z_j) \in M$, we have $x_i \neq x_j$, $y_i \neq y_j$, and $z_i \neq z_j$. Furthermore, the number of occurrences in T of any element in X, Y, or Z is at most 3. The 3DM-3 problem is to determine whether there is a matching in T of maximum size n.

Constructing a special case of DRPA. Let $\mathcal{D} = \max_{i,j} d_{ij}$, where $i \in \{1, 2, ..., p\}$ and $j \in \{p+1, p+2, ..., N\}$, i.e., $\mathcal{D}$ is the maximum distance between any data item's DG and any sensor node in DRPA. Let $\mathcal{D}$ minus the shortest distance between a data item's DG and a sensor node be the "profit" of the data item if redistributed to that sensor node. With this, the objective of DRPA is rephrased appropriately to maximizing profit, to be consistent with the maximization objective of 3DM-3. We construct the special case of DRPA as follows: a) each data item can take two distinct profit values, b) there are only two distinct data item sizes (note that the size of the data item is independent of the sensor node to which it is redistributed to, as in the original definition of the DRPA problem), and c) the storage capacities of all sensor nodes are identical.

Constructing an instance for the special case of DRPA. Now, given an instance of 3DM-3, we construct an instance of the above special case of DRPA as follows. In this special case of DRPA, there are m sensor nodes $b_1, b_2, ..., b_m$ of storage capacity 3 each, one corresponding to each of the edges $e_1, e_2, ..., e_m$ in T. For each $i \in X$, we have a data item x_i, and similarly data item y_j for $j \in Y$ and data item z_k for $k \in Z$. Besides these $3n$ data items, we have additional $m - n$ data items $u_1, u_2, ..., u_{m-n}$. Let S_i denote the size of data item i, and let $P_{i,j}$ denote the profit of data item i if redistributed to sensor node j. We assign sizes and profits to data items as follows. The sizes of data items x_i, y_j, and z_k are all set to 1 each. For a data item x_i to be redistributed to a sensor node b_l, we set $P_{x_i,b_l} = 1 + \delta$ if $i \in e_l$ and 1 otherwise, where δ is a positive constant less than 1/3. The profits of data items y_j and z_k are set similarly. For any of the $m - n$ data items u_h, its data size $S_{u_h} = 3$ and profit $P_{u_h,b_l} = 4$ for any sensor node b_l.

Now we prove that an instance of 3DM-3 contains a matching M of size n if and only if an instance of the special case of DRPA has maximum profit value of $3n(1+\delta)+4(m-n)$. We start with the "only if" direction. If T contains a matching of size n, say M, then each element $i \in X$ belongs to exactly one hyperedge in M; so does $j \in Y$ and $k \in Z$. In the instance of the special case of DRPA, this means that each data item x_i, y_j, and z_k is redistributed to a sensor node b_l with benefit of $1 + \delta$, leading to a total benefit of $3n(1 + \delta)$ and total size of $3n$ (note each data item is of size 1). This leaves available storage $3m - 3n$ at sensor nodes, which can be used to accommodate the additional $m - n$ data items u_h (each of size 3),

yielding a benefit of $4(m-n)$. Therefore the total benefit is $3n(1+\delta)+4(m-n)$. It is obvious that this benefit is the maximum possible benefit using any data redistribution scheme. To prove the "if" direction, since $3n(1+\delta)+4(m-n)$ is the maximum possible profit, it must be the case that all $m-n$ data items u_h are distributed to some sensor nodes, and all n data items x_i are distributed to some sensor nodes such that element $i \in e_l$ to yield benefit of $1+\delta$ each, so are y_i and z_k. This is possible only when the 3DM-3 instance contains a matching of size n wherein each x_i, y_i, and z_k belongs to exactly one e_l.

■

The proof technique used here is similar to that used in [Chekuri and Khanna 2005] to prove the APX-hardness of a special case of a closely related Generalized Assignment Problem [Shmoys and Tardos 1993].

REFERENCES

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

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

Chekuri, C. and Khanna, S. 2005. A ptas for the multiple knapsack problem. *SIAM J. Comput. 35,* 3, 713–728.

Erwig, M. and Hagen, F. 2000. The graph voronoi diagram with applications. *Networks 36*, 156–163.

Gao, J., Guibas, L., Milosavljevic, N., and Zhou, D. 2009. Distributed resource management and matching in sensor networks. In *Proc. of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*.

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

Goldberg, A. V. 2008. Andrew goldberg's network optimization library. http://www.avglab.com/andrew/soft.html.

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

Heinzelman, W., Chandrakasan, A., and Balakrishnan, H. 2000. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of the Hawaii International Conference on System Sciences (HICSS)*.

Hoppe, B. and Tardos, E. 2000. The quickest transshipment problem. *Math. Oper. Res. 25,* 1, 36–62.

Jackson, J. D. 1999. *Classical Eletrodynamics*, Third ed. John Wiley and Sons.

Kann, V. 1991. Maximum bounded 3-dimensional matching is max snp-complete. *Information Processing Letters 37,* 1, 27–35.

Khuller, S. and ah Kim, Y. 2003. Algorithms for data migration with cloning. In *SIAM Journal on Computing*. ACM Press, 27–36.

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

Li, K., Shen, C.-C., and Chen, G. 2010. Energy-constrained bi-objective data muling in underwater wireless sensor networks. In *Proc. of the 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2010)*. 332–341.

Li, S., Liu, Y., and Li, X. 2008. Capacity of large scale wireless networks under gaussian channel model. In *Proc. of the ACM Annual International Conference on Mobile Computing and Networking (MOBICOM)*.

Liu, C. and Cao, G. 2010. Distributed monitoring and aggregation in wireless sensor networks. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*.

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

Luo, L., Huang, C., Abdelzaher, T., and Stankovic, J. 2007. Envirostore: A cooperative storage system for disconnected operation in sensor networks. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM).*

Ma, M. and Yang, Y. 2008. Data gathering in wireless sensor networks with mobile collectors. In *Proc. of the IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2008).* 1–9.

Martinez, K., Ong, R., and Hart, J. 2004. Glacsweb: a sensor network for hostile environments. In *Proc. of the Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON).*

Mathioudakis, I., White, N. M., and Harris, N. R. 2007. Wireless sensor networks: Applications utilizing satellite links. In *Proc. of the IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2007).* 1–5.

Mathur, G., Desnoyers, P., Ganesan, D., and Shenoy, P. 2006. Ultra-low power data storage for sensor networks. In *Proc. of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN).*

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

Nuggehalli, P., Srinivasan, V., and Chiasserini, C.-F. 2003. Energy-efficient caching strategies in ad hoc wireless networks. In *Proc. of MOBIHOC 2003.*

Orlin, J. 1990. A faster strongly polynomial minimum cost flow algorithm. *Operations Research 41,* 2, 338–466.

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

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

Shah, R. C., Roy, S., Jain, S., and Brunette, W. 2003. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *Proc. of the IEEE Workshop on Sensor Network Protocols and Applications (SNPA).*

Shmoys, D. and Tardos, E. 1993. An approximation algorithm for the generalized assignment problem. *Mathematical Programming 62,* 3, 461–474.

Siegel, S. F. and Siegel, A. R. 2008. Madre: The memory-aware data redistribution engine. In *European PVM/MPI Users' Group Meeting (PVM/MPI).*

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

Syed, A. A., Ye, W., and Heidemann, J. 2008. T-lohi: A new class of mac protocols for underwater acoustic sensor networks. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM).*

Tardos, E. 1985. A strongly polynomial minimum cost circulation algorithm. *Combinatorica 5,* 3, 247–255.

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

Vasilescu, I., Kotay, K., Rus, D., Dunbabin, M., and Corke, P. 2005. Data collection, storage, and retrieval with an underwater sensor network. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys).*

Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., and Welsh, M. 2006. Fidelity and yield in a volcano monitoring sensor network. In *Proc. of the USENIX Symposium on Operating Systems Design and Implementation (OSDI).*