Computer Science Department

College of Natural and Behavioral Sciences

California State University, Dominguez Hills

DAO: Data Aggregation and Offloading in Sensor Networks

Basil Alhakami
Computer Science Department
California State University Dominguez Hills

THESIS: DAO: DATA AGGREGATION AND OFFLOADING IN SENSOR NETWORKS

AUTHOR: BASIL ALHAKAMI

APPROVED

____
DR. Mohsen Beheshti
Department Chair and Committee Member

____
DR. Jianchao "Jack" Han
Graduate Program Coordinator and Committee Member

____
DR. Bin Tang
Committee Chair and Advisor

Acknowledgments

This work was supported in part by NSF Grant CNS-1116849.

I would like to thank Dr. Bin Tang for his support. This thesis would not be possible without his guidelines.

I would like to also thank Dr.Jack Han and Dr. Mohsen Behesthi for their valuable advice.

Finally I dedicate my thesis to my father, mother, my sister Hind, and to my brothers: Mohammed, Khalid, and Mohannad.

Abstract

When sensor networks are deployed in an inaccessible or inhospitable region, or under extreme weather, it is not usually viable to install a long-term base station in the field to collect data. The generated sensory data is therefore stored inside the network first. However, when more data is generated than available storage spaces in the entire network can possibly store, data loss arises. This problem is referred to as the *overall storage overflow* in sensor networks. In this thesis, a new algorithmic problem called the *data aggregation and offloading problem (DAO)*, is used to overcome overall storage overflow. By taking advantages of data spatial correlation that commonly exists among sensory data, the DAO aggregates overflow data to the size that can be accommodated by the available storage capacity in the network, and then offloads the aggregated data inside the network to be stored. The goal of the DAO is to minimize energy consumption incurred in the above data aggregation and data offloading process. The research shows that the DAO is NP-hard. To solve the DAO, a naive two-stage solution is first proposed, which is referred to as DAO-N, wherein data offloading strictly follows data aggregation. A more unified method that is based upon data replication techniques, referred to as DAO-R, is proposed in order to further reduce energy consumption. Specifically, a series of data replication algorithms is designed to integrate data aggregation and data offloading. A sufficient condition to solve DAO-R optimally is also provided. The research shows via extensive simulations that DAO-R outperforms DAO-N by approximately 20% in terms of energy consumption, under different network parameters.

Keywords – data aggregation, data offloading, overall storage overflow, sensor networks, energy efficiency

TABLE OF CONTENTS

Chapter	Page
1. INTRODUCTION	1
1.1 Background and motivation	1
1.2 Thesis Organization	3
2. RELATED WORK	4
3. DATA AGGREGATION AND OFFLOADING PROBLEM (DAO)	6
3.1 Network Model and Data Correlation Model	7
3.2 Problem Formulation of DAO	9
3.3 NP- Hardness of DAO	11
4. DAO-N A NAIVE TWO-STAGE APPROACH	12
4.1 Data Aggregation Approximation Algorithm	12
4.2 Data Offloading Algorithm	13
4.3 Limitations of Naïve Two-Staged Approach	14
5. DAO-R INTEGERATING DATA AGGREGATION AND DATA OFFLOADING VIA REPLICATION	16
5.1 Problem Formulation of DOA-R	16
5.2 Global Replication Algorithm	18
5.3 Localized Replication Algorithm	20
6. PERFORMANCE EVALUATION	24
7. CONCLUSION AND FUTURE WORK	28

Chapter 1

INTRODUCTION

1.1 Background and Motivation

In this thesis, the focus is on some emerging sensor networks, such as underwater sensor networks [12], wind and solar harvesting [10, 18], and ecological monitoring [16, 22]. A common characteristic of such networks is that they are all deployed in inaccessible or inhospitable regions, or under extreme weather, to constantly collect data from the physical environments for a long period of time. Due to the inaccessible and hostile environments, it is not viable to deploy base stations (with power outlets) to collect data in or near the sensor fields. Therefore, data generated have to be stored inside the sensor network for some period of time and then be collected by periodic visits of robots or data mules [9], or by low-rate satellite links [17]. Meanwhile, storage is still a serious resource constraint of sensor nodes, despite the advances in energy-efficient flash storage [4, 19] with good compression algorithms (data is compressed before stored) and good aging algorithms (fidelity of older data is reduced to make space for newer data). As a consequence of this resource constraint, the massive sensory data could soon overflow data storage of sensor nodes and cause data loss. Below, two levels of data overflow and their corresponding solutions are outlined:

- Node storage overflow. The first level of data overflow is *node storage overflow*, wherein some data-generating sensor nodes deplete their own storage spaces, causing data loss. These sensor nodes with depleted storage spaces, while still generating data, are referred to as *data nodes*. The newly-generated data that can no longer be stored in data nodes is called *overflow data*. The solution to avoid such data loss is simple: the overflow data is offloaded to other nodes with available storages (referred to as *storage nodes*). Different data offloading techniques have been proposed with the goal of either minimizing the total energy consumption during data offloading [20], or maximizing the minimum remaining energy of storage nodes to prolong network lifetime [8], or offloading the most useful information, considering data could have different priorities [25]. However, these techniques do not address the second level of data overflow, which is *overall storage overflow,* explained below.

- <u>Overall storage overflow</u>. This happens when the total size of the overflow data is larger than the total size of the available storage in the network. Data offloading alone cannot solve this problem; discarding data becomes inevitable if no action is taken. This is a more severe problem compared to node storage overflow. For the large amount of overflow data generated in the sensor networks, figuring out how to store as much useful information as possible using the available storage of the network becomes a new challenge.

In order to achieve fine-grain monitoring in a wide variety of applications, it usually requires dense sensor nodes deployment in wireless sensor networks. Due to the high density of nodes, spatially proximal sensor observations are highly correlated [11]. The spatially redundant or correlated data provides an opportunity to solve the aforesaid overall storage overflow problem—it allows for aggregation and reduces the size of overflow data without sacrificing information loss. Formulating a new algorithmic problem, referred to as *Data Aggregation and Offloading (DAO)* exploits such advantages brought by spatial correlation among sensor data. First is a naive, two-stage approach called DAO-N. This approach treats data aggregation and data offloading as two independent stages, and solves each separately. That is, it first aggregates and reduces the size of overflow data such that it can be accommodated by the available storage in the network, then it offloads the data into the network. In particular, an approximation algorithm [21] is presented to solve this data aggregation problem. The data offloading stage can be solved optimally, by showing that it is equivalent to a minimum cost-flow problem [20]. Second, demonstrating that solving each stage independently and combining the results together does not necessarily achieve best performance. Third, a unified approach, called DAO-R, is presented to leverage the synergies existing between data aggregation and data offloading. Specifically, a series of data replication algorithms is designed, wherein data is replicated in the way of data aggregation. The novelty of DAO-R is that replicating data along aggregation paths achieves the effect of "offloading" overflow data to storage nodes, without introducing extra energy cost. A sufficient condition that solves DAO-R optimally is also given. Finally, the research shows via extensive simulations that DAO-R outperforms DAO-N by approximately 20% in terms of energy consumption, under different network parameters.

1.2 Thesis Organization.

The rest of the thesis is organized as follows: Chapter 2 discusses state-of-the-art and related work, to give context of the contribution of the work. In Chapter 3, the overall storage problem is introduced, presenting the network, data correlation, and energy models, and the formulation of the data aggregation and offloading (DAO) problem. Chapter 4 presents DAO-N, a naive, two-stage approach for DAO. In Chapter 5, a more energy-efficient and unified approach to solve DAO called DAO-R is presented, by designing a suite of data replication algorithms to integrate data aggregation and offloading. Also provided is an efficient condition under which DAO-R can be solved optimally. In Chapter 6, all of the different algorithms under different network dynamics are compared, and the simulation results are discussed. Chapter 7 concludes the thesis with possible future research.

Chapter 2

RELATED WORK

Figuring out how to preserve data in sensor networks in the absence of the base station has become part of active research in recent years. In particular, Tang et al. [20] addressed the energy-efficient data redistribution problem in data-intensive sensor networks. Hou et al. [8] studied how to maximize the minimum remaining energy of the nodes that finally store the data, in order to store the data for long period of time. Xue et al. [25] considered that sensory data from different source nodes have a different importance, and they studied how to preserve data with the highest importance. However, none of the works addresses the overall storage overflow problem. Tang and Ma [21] recently studied the overall storage overflow problem by solving the problem of aggregating data and reducing its size so that it can be accommodated by the available storage. However, important research problems such as how to offload the aggregated data to storage nodes and how to integrate data aggregation and data offloading to further save energies are not addressed, which is the topic of this thesis. Ganesan et al. [4] adopted data aggregation techniques to tackle the storage constraint of sensor networks. They proposed wavelet compression techniques to construct summaries for data at different spatial resolutions, as well as a progressive aging scheme, wherein older data gets more aggressively summarized to save storage space for newer data. The summarization is based on a hierarchical grid-based overlay, in which summaries at each higher level of the hierarchy encompass larger spatial scales but are lossier overall. In contrast, our approach does not rely upon any hierarchy of overlays. Traditional data aggregation in sensor networks collects sensor data by combining the data from different sensor nodes on the way to the base station, in order to eliminate redundancy and to reduce energy consumption during data collection. As a result, the underlying routing structures for data aggregation are usually trees rooted at the base station. Data aggregation techniques have been designed for different purposes. Some are used to maximize the network lifetime (the time until the first node depletes its energy) [15, 23], some are used to minimize the total energy consumption or communication cost [13, 14], and some are used to reduce the delay of data gathering [24]. In contrast, the overall storage overflow problem studied in this thesis takes place when the base station does not exist and there is not enough storage to store all the overflow data. Consequently, data aggregation in the DAO has a very different goal compared to traditional data aggregation. It is to aggregate the overflow data so that their size can be reduced and accommodated by the storage spaces available in the network, in order to prevent data loss caused by overall storage overflow.

Chapter 3

DATA AGGREGATION AND OFFLOADING PROBLEM (DAO)

In this chapter, the DAO is illustrated with an example. Presented next is its network model, data spatial correlation model, and energy model. Finally, the DAO is formally formulated.

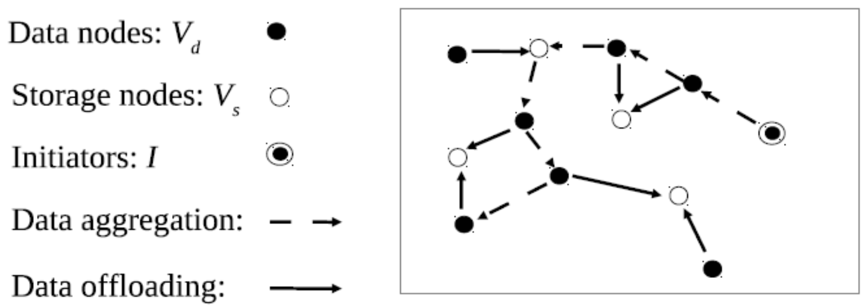


Fig. 1. An illustration of DAO.

The sensor network studied consists of *data nodes* (with overflow data) and *storage nodes* (with available storage spaces), as shown in Figure 1. The total size of the overflow data from the data nodes exceeds the total size of the storage spaces from the storage nodes, resulting in overall storage overflow. To start aggregating data, one or multiple data nodes (called *initiators*) send their whole overflow data to the other data nodes. When a data node (called an *aggregator*) receives the data, it aggregates its own overflow data (each aggregator can aggregate its data only once). After that, the aggregator forwards the initiators' overflow data to another data node, which then becomes an aggregator and aggregates its own overflow data, and so on and so forth. This continues until enough aggregators are visited such that the total size of the overflow data after aggregation equals to or is slightly less than the total available storage in the network. At this point, there is zero amount of overflow data on each initiator; the last aggregator being visited by each initiator has both its own aggregated data and the whole data from the initiator, and all other aggregators have their own aggregated overflow data. If a data node is not involved in data aggregation (i.e., not an initiator and not an aggregator), its overflow data is not aggregated. After aggregation, all the overflow data (aggregated or not) are then offloaded to storage nodes. Figure 1 shows both data aggregation and data offloading. Our goal is to minimize the total energy consumption in the entire process.

3.1 Network Model and Data Correlation Model

The sensor network is represented as a graph$G(V, E)$, where $V = \{1, 2, \dots, |V|\}$ is the set of $|V|$ sensor nodes uniformly distributed inside the network, and E is the set of $|E|$ edges. All the nodes have the same transmission range; an edge exists between two nodes if their distance is within the transmission range. There are p data nodes (denoted asV_d); each has R bits of overflow data. The other $|V|$ - p nodes are storage nodes, denoted as V_s, each of which has m bits of available storage spaces (this work can be extended to cases of varying data and storage size).

The data correlation model adopted is proposed in [3]. $H(X|Y)$ denotes the conditional entropy of a random variable X given that random variable Y is known. Overflow data at data node i is represented as an entropy $H(i) = R$ bits if no side information is available from other data nodes; $and\ H(i \mid j_1, \dots, j_p) = r \leq R$ bits, $j_k \in V_d \wedge j_k \neq i, 1 \leq k \leq p,$ if data node i has available side information from at least another data node. In other words, if a data node receives data from at least another data node, its overflow data can be aggregated and reduced from R to r. This correlation model is adopted because a) it is a simple and distributed coding strategy, making it easy to implement in large-scale sensor network application and b) it is a realistic model that approximates the case where the correlation function between two nodes decreases with their distance [3].

First Order Radio Energy Model [7]. For node u sending R-bit data to its one-hop neighbor v over their distance $l_{u,v}$, the transmission energy cost at u is $E_t(R,\ l_{u,v}) = E_{elec} \times R + \epsilon_{amp} \times R \times l_{u,v}^2$, the receiving energy cost at v is $E_r(R) = E_{elec} \times R$. Here, $E_{elec} = 100nJ/bit$ is the energy consumption per bit on the transmitter circuit and receiver circuit, and $\epsilon_{amp} = 100pJ/bit/m^2$ calculates the energy consumption per bit on the transmit amplifier. Let $w(R,u,v) = E_t(R,l_{u,v}) + E_r(R)$. Let $W = \{v_1,v_2,\ldots,v_n\}$ be a sequence of n nodes with $(v_i,v_{i+1}) \in E, 1 \le i \le n-1$ and $v_1 \neq v_n$. If all the nodes in W are distinct, W is a *path*; otherwise, it is a *walk*. Let $c(d,W) = \sum_{i=1}^{n-1} w(d,v_{i,}\, v_{i+1})$ denote the energy consumption of sending d-bit of data along W. Next, the differences of energy consumptions between data aggregation and data offloading stages are explained. In the data aggregation stage, the route that the R-bit overflow data at each initiator traverses could be either a path or a walk, because enough of a number of aggregators needs to be visited in order to reduce overflow data size. Let $c(R,W)$ denote the *aggregation cost* of R-bit data traversing along path/walk W starting from its initiator. In the data offloading stage, however, the route that the overflow data traverses is always a path, in order to minimize the total offloading cost. Besides, in the data offloading stage, each offloaded overflow data unit is not necessarily in sizes of R or r. Instead, for the purpose of energy efficiency, the overflow data at each data node can be split into smaller units, each of which can be offloaded to different storage nodes. Let the size of each smaller unit be x-bit, then $c(x,W)$ is the *offloading cost* of offloading this x-bit data from its data node v_1 to a storage node v_n, along path W. It is assumed that there exists a contention-free MAC protocol to avoid overhearing and collision (e.g. [2]), so that the energy consumption contains only two parts: transmitting data and receiving data.

Feasible Overall Storage Overflow. Feasible overall storage overflow refers to the conditions that a) there is an overall storage overflow, b) there are enough numbers of aggregators to visit in order to reduce the data size, and c) the data after aggregation can fit in the available storage in the network. For a), we have $p \times R > (|V| - p) \times m$ which is $p > \frac{|V|m}{m+R}$. For b) and c), since the size of overflow data that needs to be reduced is $p \times R > (|V| - p) \times m = p \times (R + m) - |V| \times m$, and visiting one aggregator reduces its overflow data size by (R - r), the number of needed aggregators, denoted as q, is

$$q = \left\lceil \frac{p \times R - (|V| - p) \times m}{R - r} \right\rceil = \left\lceil \frac{p \times (R + m) - |V| \times m}{R - r} \right\rceil. (1)$$

Among all the p data nodes, since at least one needs to be an initiator (therefore cannot be an aggregator), there is $q \leq p - 1$. Plug in Equation 1, and $p \leq \left\lfloor \frac{|V|m - R + r}{m+r} \right\rfloor$ therefore, the valid range of p for feasible overall storage overflow is

$$\frac{|V|m}{m + R} < p \leq \left\lfloor \frac{|V|m - R + r}{m + r} \right\rfloor. (2)$$

Given p and q, at most (p - q) data nodes can be selected as initiators.

3.2 Problem Formulation of DAO.

Let $D = \{D_1, D_2, \ldots, D_{|D|}\}$ denote the set of $|D|$ overflow data *after* aggregation, each is x-bit. Let $s(i)$, where $1 \leq i \leq |D|$, denote the data node of data unit D_i. Given an instance of feasible overall storage overflow, the DAO decides:

- a set of l $(1 \leq l \leq (p - q))$ initiators $I \subset V_d$, and a corresponding set of l *aggregation paths/walks*: $W_1^a, W_2^a, \ldots, W_j^a$,, where W_j^a $(1 \leq j \leq l)$ starts from a distinct initiator $Ij \in I, and$ $\left|\bigcup_{j=1}^{i}\{W_j^a - \{Ij\} - Gj\}\right| = q$ (G_j is the set of storage nodes in W_j^a), and

- an offloading function $o : D \rightarrow V_s$, to offload data unit $D_i \in D$ from its data node $s(i) \in V_d$ to storage node $o(i) \in V_s$; or equivalently, a set of $|D|$ offloading paths: $W_1^o, W_2^o, \ldots, W_{|D|}^o$, where W_i^o, $(1 \leq i \leq |D|)$ starts from $s(i)$ and ends with $o(i)$.

The goal of the DAO is to minimize the *total energy cost* in aggregation and offloading:

$$C_{total} = \sum_{i \leq j \leq l} c(R, W_j^a) + \sum_{i \leq j \leq |D|} c(x, W_j^o), \quad (3)$$

under the constraint that the size of overflow data offloaded to any storage node cannot exceed its available storage capacity $|\{j | 1 \leq j \leq |D|, o(j) = i\}| \times x \leq m, \forall i \in V_s$.

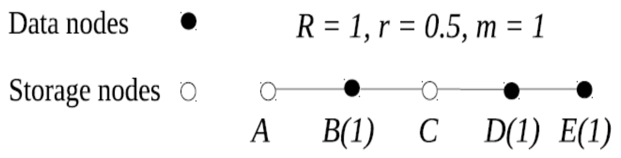


Fig. 2. A sensor network with overall storage overflow problem.

EXAMPLE 1: Fig. 2 illustrates the overall storage problem with a linear sensor network of five nodes. Nodes *B*, *D*, and *E* are data nodes, while *A* and *C* are storage nodes. Numbers in parentheses are the overflow data sizes *R*. The energy consumption along any edge is one per unit of data. There are a total of 3 units of overflow data, while there are only 2 units of available storage, causing overall storage overflow. Number of aggregators q is calculated as 2 using Equation 1. This leaves one data node to be an initiator. Chapter 4 shows how to select the initiator and corresponding aggregation path to solve this overall storage overflow problem. □

3.3 NP-Hardness of DAO.

The DAO is NP-hard, since its constituent data aggregation problem itself is NP-hard [21]. Therefore, a time-efficient approximation algorithm and heuristic algorithm is designed to solve the DAO. Chapter 4 presents a naive, two-stage solution to solve the DAO and to show its limitation. Chapter 5 proposes a more unified approach as well as more energy efficiency which employs data replication techniques to integrate data aggregation and offloading.

Chapter 4

DAO-N: A NAÏVE TWO-STAGE APPROACH

A naive approach to solve the DAO is to solve data aggregation and data offloading as two separate and independent problems, and then combine the solutions (i.e., the energy cost of the DAO is the sum of the costs of data aggregation and data offloading). It first aggregates overflow data to the size that can be accommodated by the available storage spaces; then, it offloads the overflow data from data nodes to storage nodes. This naive two-stage approach is referred to as DAO-N. Below the algorithms solving data aggregation and data offloading, are presented, respectively.

4.1 Data Aggregation Approximation Algorithm

It has been proven [21] that the data aggregation problem in a general graph topology is NP-hard. Below, an approximation algorithm, which yields energy cost that is at most $\left(2-\frac{1}{q}\right)$ times of the optimal [21] is presented.

Algorithm 1: Data Aggregation Approximation Algorithm.

1). Transform the sensor network graph $G(V,E)$ to its *aggregation graph* $G'(V',E')$, defined below. V' is the set of p data nodes in V, i.e. $V' = V_d$. For any two data nodes $u,v \in V_d$ in G, there exists an edge $(u,v) \in E'$ in G' if and only if all the shortest paths between u and v in G do not contain other data nodes. For each edge $(u,v) \in E'$, its weight $w(u,v)$ is the cost of the shortest path between u and v in G.

2). Create a set S containing all the edges in E' in non-decreasing order of their weights.

3). Create a forest F of $|V|$ trees, initially each tree is one of the $|V|$ nodes.

4). Starting from the first edge in *S*, if that edge connects two different trees, add it to F and combine two trees into a single tree. This repeats for q times.

5). Replace each edge (u, v) in *F* with a shortest path between *u* and *v* in *G* (choose one randomly if there are multiple).

6). For each connected component of the resulted *F*, if it is linear, it starts from one end (the initiator) and visits the rest of the nodes exactly once; if it is a tree, it does the following: finding an edge (u, v) with maximum weight in the tree (tie is broken randomly), which has three parts; T_u, (u, v), and T_v, it starts from u (the initiator) and visits all the nodes in T_u in a sequence following depth-first-search (DFS) and comes back, then visits *v*, from where it visits all the nodes in T_v in a sequence following DFS.

Applying the above approximation algorithm to the sensor network in Fig. 2, it obtains two data aggregation solutions. One is that data node *B* is an initiator and the aggregation path is *B*, *C*, *D*, and *E*, as shown in Fig. 3(a). It also shows the sizes of overflow data at each data node after aggregation (but before offloading). In particular, there is no overflow data left at initiator *B*, and there are 0.5 and 1.5 units of data at data nodes *D* and *E* respectively. The aggregation cost is 3, which happens to be optimal in this small network.

4.2 Data Offloading Algorithm

Data offloading is to offload the overflow data from its data node to the storage node, since there are enough storage spaces available to store the overflow data after data aggregation. The goal of data offloading is to minimize the energy cost incurred in the offloading process. Tang et al. [20] show that by transforming the sensor network graph $G(V, E)$ into a flow network, the data offloading problem is equivalent to the minimum cost-flow problem [1], which is solvable optimally in polynomial. In this thesis, the scaling push-relabel algorithm proposed and implemented in [5,6], is adopted, with a time complexity of $O(|V|2|E|log(|V|C))$. Here $C = \max\{\frac{R+r}{x}, \frac{m}{x}\}$ is the maximum capacity of an edge in the transformed graph.

Fig. 3(b) shows the data offloading solution that follows data aggregation in Fig. 3(a). It shows that 0.5 unit of data at *D* and 0.5 unit of data at *E* are offloaded to storage node *C*, while 1 unit of data at *E* is offloaded to storage node *A*, with an offloading cost of 5.5.

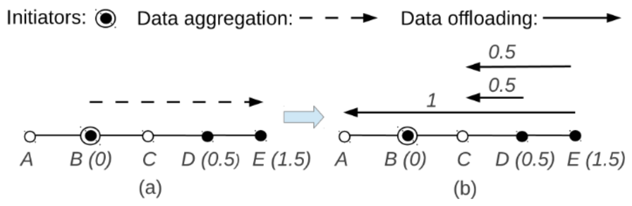


Fig. 3. One naive two-stage solution with *B* being the initiator. (a) Data aggregation stage: values in parentheses are sizes of overflow data after aggregation. (b) Data offloading stage: values on the arrowed lines are sizes of overflow data that is offloaded from its data node to a storage node.

4.3 Limitations of Naive Two-Stage Approach.

Another naïve two-stage solution is shown in Fig. 4, wherein data node *E* is the initiator and the aggregation path is *E*, *D*, *C*, and *B*. In this case, the aggregation cost is again 3. However, the offloading cost is 2, which is a 64% improvement compared to 5.5 in Fig. 3(b). Therefore, even though the solution in Fig. 3 independently solves each of the data aggregation and data offloading nicely (one with approximation algorithm and the other optimal algorithm); the combined solution may not give the best result.

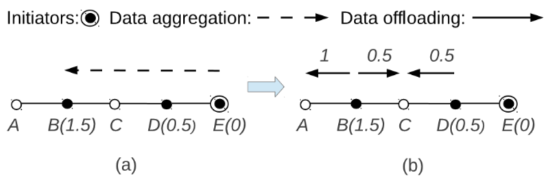


Fig. 4. Another naive two-stage solution with *E* being the initiator.

Furthermore, even though Fig. 4 gives optimal combined total energy cost of data aggregation and offloading, its performance can be further improved. The key observation here is that while aggregating data, it can also replicate data along the aggregation paths, since replicating data does not introduce extra energy consumption. As a result of replicating, less data needs to be offloaded in the data offloading stage. Fig. 5 shows that when initiator *E* sends its one unit data passing storage node *C*, it replicates half of the data and stores it at *C*. Therefore, next in the data offloading stage, node *B* only needs to offload the other half unit of *E* to *A* (combined with its own 0.5 unit after aggregation, *B* actually offloads one unit to *A*).The offloading cost is 1.5, a 25% of improvement compared to offloading cost of 2 in Fig. 4(b).

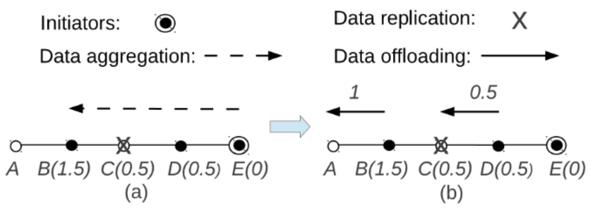


Fig. 5. Illustrating the DAO with data replication. 0.5 unit of initiator *E*'s data is replicated and stored at *C* in the data aggregation stage. Thereafter in the data offloading stage, node *B* does not need to offload this part of data.

Chapter 5

DAO-R: INTEGRATING DATA AGGREGATION AND DATA OFFLOADING VIA REPLICATION

In this chapter, the DAO-R that integrates data aggregation and offloading is formulated, and solved by designing two data replication algorithms. It is assumed that *l* data aggregation paths $W_1^a . W_2^a, \dots, W_l^a$ have already been found using Algorithm 1 in chapter 4. As in chapter 3, it is assumed that the overflow data consists of small units, each of which is *x*-bit and different units can be offloaded to different storage nodes. The overflow data that needs to be offloaded after data aggregation falls into one of the three categories:

- $D' = \bigcup_{j=1}^{l} D_j'$: The overflow data of all the initiators.D_j' is the R amount of overflow data of I_j an initiator of W_j^a $(1 \leq j \leq l)$. Note that each initiator's overflow data has been transmitted to the last aggregator of each aggregation path after data aggregation.
- D'': The overflow data of all the aggregators, each having r amount of overflow data.
- D''': The overflow data that are not aggregated and are not on any aggregation path. D''' is empty if all the data nodes are on some aggregation paths.

5.1 Problem Formulation of DAO-R

When the *R* amount of overflow data traverses each aggregation path starting from its initiator, it can replicate part or all *R* on storage nodes along the path. The data replication algorithms decide for each aggregation path W_j^a

- the end node I_j that serves as initiator,
- a subset of D_j' , denoted as D_j , of data units to replicate,

- a *replication function* $r : Dj \rightarrow V_S \cap W_j^a$, to replicate and store a data unit in D_j at a storage node in W_j^a , when R amount from I_j traverses along $W_j^{a.}$

under the constraint that the total size of replicated data on any storage node along any aggregation path cannot exceed this node's available storage capacity: $|\{k|k \in D_j, r(k) = i\}| \times x \leq m, \forall i \in V_s \cap W_j^a$. Recall that D is the entire set of data units to be offloaded after data aggregation. With D_j replicated and stored on W_j^a, the rest of the $D_j' - D_j$ amount still needs to be offloaded from the last aggregator of W_j^a in the data offloading stage. Therefore,

$$D = \bigcup_{j=1}^{l} (D_j' - D_j) D'' \cup D''' = (D' - \bigcup_{j=1}^{l} D_j) \cup D'' \cup D'''$$

Let $s(j)$ denote the data node of any data unit $D_j \in D$. The data offloading algorithm is to decide an offloading function $o : D \rightarrow V_s$, to offload $D_i \in D$ from its data node $s(i) \in V_d$ to storage node $o(i) \in V_s$. Equivalently, the data offloading algorithm is to decide a set of $|D|$ offloading paths: $W_1^o, W_2^o, \dots, W_{|D|}^o$, where W_j^o ($1 \leq j \leq |D|$) starts from $s(j)$ and ends with $o(j)$, to minimize the offloading cost:

$$C_{off} = \sum_{1 \leq j \leq |D|} c(x, W_j^o), (4)$$

under the constraint that the size of overflow data offloaded to any storage node in the network cannot exceed its available storage capacity: $|\{j|1 \leq j \leq |D|, o(j) = i\}| \times x \leq m, \forall i \in V_s$. In DAO-R, since all the aggregation paths are given, the total aggregation cost is the same whether replication takes place or not. Therefore the offloading cost needs to be minimized, which can be solved by the minimum cost-flow algorithm given optimally in polynomial time [5,6].

Selecting initiator for each aggregation path. After aggregation, among all data nodes on a particular aggregation path, the initiator has the least amount of overflow data (zero), the last aggregator has the most (R + *r*), while others have the same *r* amount of overflow data. Therefore, having more available storage spaces around the last aggregator would make the data offloading next more energy-efficient. For example, in Fig. 2, since *B* has two neighboring storage nodes while *E* has zero, *E* is selected as the initiator.

5.2 Global Replication Algorithm.

Once the initiator is selected for each aggregation path, it begins the data aggregation and replication process. The global replication algorithm works as follows: first, it offloads the *r* amount of data at all the aggregators and the overflow data that are not aggregated (line 1), then for each aggregation path, it finds its available spaces (line 3-7). The amount to be replicated is the smaller of *R* and the size of the available spaces (line 8). Next, while the *R* amount of data from the initiator is traversing along the path performing the data aggregation, it replicates this amount (line 9-16). Finally, it offloads each initiator's overflow data that has not been offloaded from the last aggregator of each path (line 18). Since it uses the minimum cost-flow algorithm to find the available spaces to replicate, Algorithm 2 takes a global perspective and is therefore referred to as *Global*. For ease of presentation, in algorithms below, $v = mc(O, G)$ means running the minimum cost-flow algorithm on $G(V, E)$ to offload a set of data units O from its belonged aggregators, yielding a minimum energy cost of v.

Algorithm 2: Global Data Replication Algorithm.

Input: All aggregation paths in $G(V, E)$: W_j^a $(1 \le j \le l)$

Output: C_{off}

0. **Notations:**

u: a node in W_j^a;

$u.next$: the next node of u in W_j^a;

z: total size of data in D_j that are not yet replicated;

$avail(u)$: amount of available storage at node u;

$avail_j$: amount of available storage at node W_j^a;

```
1.  $C_{off} = mc(D'' \cup D''', G)$;
2.  for each $W_j^a$ $(1 \leq j \leq l)$
3.   $avail_j = 0, u = I_j.next$;
4.  while ($u$ is not the last aggregator on $W_j^a$)
5    $avail_j = avail_j + avail(u)$;
6.      $u = u.next$;
7.      end while;
8.      $|D_j| = min\{avail_j, R\}$;
9.      $u = I_j.next, z = |D_j|$;
10.     **while** $(z > 0)$
11.     **if** $(u \in V_s)$
12.     Replicate $avail(u)$ amount at $u$;
13.     $z = z - avail(u)$;
14.     **end if**;
15.     $u = u.next$;
```

```
16.            **end while**;
17.            **end for;**
18.            $C_{off} = C_{off} + mc(D' - \bigcup_{j=1}^{l} D_j \ , G)$;
19. **RETURN** $C_{off}$.
```

Time complexity. The minimum cost-flow algorithm takes $O(|V|^2|E|log(|V|C))$, with. $C = \max\left\{\frac{R+r}{x}, \frac{m}{x}\right\}$. Since each of the l $(l = O(|E|))$ aggregation paths can not have more than $|V|$ nodes, finding available storages and replicating data along each aggregation path takes $O(|V|)$. It takes $O(|E| \times |V|)$ for all the aggregation paths. Therefore, the time complexity of Algorithm 2 is $O(|V|^2|E|log(|V|C))$.

Theorem 1: In Algorithm 2, if $\forall\, 1 \le j \le l,\ avail_j \ge R$, then C_{off} is minimum.

Proof: Recall $D = \bigcup_{j=1}^{l}(D_j' - D_j) \cup D'' \cup D'''$. Now if $\forall\ 1 \le j \le l, avail_j \ge R$ then $|D_j| = min\{avail_j, R\} = R$. Recall that D_j is exactly this R amount of overflow data of I_j. Therefore, $D_j' = D_j, \forall\ 1 \le \mathrm{j} \le \mathrm{l}$. $\mathrm{D} = D'' \cup D'''$, which is the minimum amount of data that must be offloaded. Therefore $Coff = mc(D'' \cup D''', G)$ is minimum.

Therefore $C_{off} = mc(D'' \cup D''', G)$ is minimum.

5.3 Localized Replication Algorithm.

The following definition is presented.

Definition 1: (Demand Number $d(u)$ of Storage Node u) For any storage node u on any aggregation path, let $N(u)$ be all its one-hop neighbors. For each data node $v \in N(u) \cap V_d$, let $s(v)$ denote number of v's one-hop neighbors that are storage nodes. Then $d(u) = \sum_{v \in N(u) \cap Vd} \frac{1}{s(v)}$ □

Note that $s(v) \neq 0$ since v has at least one neighboring storage node u. The idea behind $d(u)$ is that the more number of data nodes surrounding u and the less storage nodes surrounding such data nodes, then more likely u will be used to store the overflow data from those data nodes.

Therefore, less of the initiator's data D_j' $on\ u$ should be replicated.

The algorithm works as follows: it first calculates the demand number of each storage node on the aggregation path, and then sorts the storage nodes in ascending order of their demand numbers. Next, it calculates the amount of data that is to be replicated at the storage node with the smallest demand number, say u, as $min(R/d(u), R)$. If not, the whole part of R is replicated as well, as not all the storage nodes on this path have been considered. It then calculates the amount to be replicated on the storage node with the second-smallest demand numbers, and so on. It stops if either the whole part of R is replicated or if all storage nodes on the path are considered for replication. It is a localized algorithm since it works on each aggregation path one by one, and figures out the replication strategy for each storage node on the aggregation path based on demand number.

Algorithm 3: Localized Data Replication Algorithm.

Input: All aggregation paths in *G (V,E):* W_j^a $(1 \leq j \leq l)$

Output: C_{off}

0. **Notations**

   $S_j = \{S_1^j, S_2^j, \ldots, S_{K_j}^j\}$: the set of K_j $(K_j \geq 0)$ storage

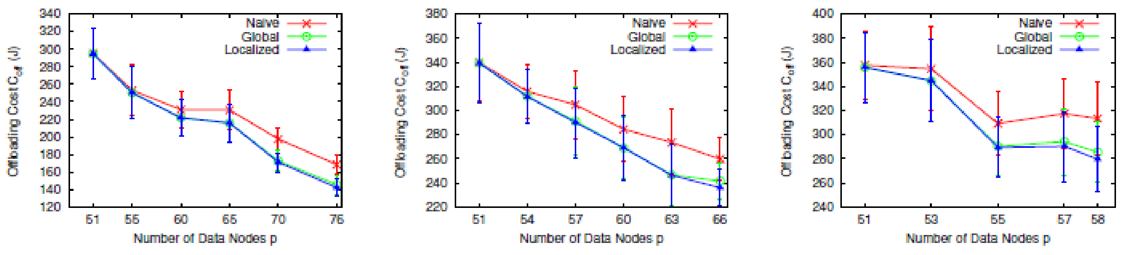
```
Nodes on $W_j^a$;
$i$: index for storage nodes.
$z$: total size of data in $D_j$ that are not yet replicated;
1.  **for** each $W_j^a$ $(1 \leq j \leq l)$
2.      **for** $u \in S_j$. calculate $d(u)$;
3.      Sort $S_j$ in assending order of $d(u)$;
4.      Let $d(S_1^j) \leq d(S_2^j) \leq \cdots \leq d(S_k^j)$,WLOG;
5.      $z = R. i = 1$;
6.      **while** $(z > 0 \ \wedge i \leq k)$
7.          Replicate min($\frac{R}{d(S_i^j)}, R$) amoount data $D_j$ on $S_i^j$;
8.          $z = z - \frac{R}{d(S_i^j)}$;
9.          $D_j' = D_j' - D_j$ ;
10.         $i++$;
11.         **end while;**
12.     **end for;**
13. **end for;**
14. $D = \bigcup_{j=1}^{l}(D_j' - D_j) \cup D'' \cup D'''$
15. $C_{\boldsymbol{off}} = mc(D, G)$;
16. $\boldsymbol{RETURN}\ C_{\boldsymbol{off}}$.
```

Time complexity. For each aggregation path, finding the demand number for a storage node takes $O(|E|^2)$ (assuming an adjacency list graph data structure), sorting the storage nodes takes $|V\ |log|V\ |$, and traversing each aggregation path takes $O(V\ )$. Each of the l aggregation paths could have at most $|E|$ edges. Therefore it takes $O(|E|^3 + |V\ |log|V\ | + V\ )\ =\ O(|E|^3))$ for all the aggregation paths. The minimum cost-flow algorithm takes $O(|V\ |^2|E|log(|V\ |C))$, with $C = \max\left\{\frac{R+r}{x}, \frac{m}{x}\right\}$.Therefore the total time complexity of Algorithm 3 is $O(|E|^3)$.

Chapter 6

PERFORMANCE EVALUATION

The performances of different DAO algorithms proposed in this thesis have been compared, via a naive two-stage algorithm (Naive), the Global Replication Algorithm (Global), and the Localized Replication Algorithm (Localized). In the simulation, 100 sensors are uniformly distributed in a region of 1000*m*×1000*m* square. The transmission range of sensor nodes is 250*m*. Unless otherwise mentioned, *R* = *m* = 512*KB* (the ratio of *R/m* is varied to 5 and 10, with *m* = 512*KB* while *R* = 2560*KB* and 5120*KB*, respectively). In all plots, each data point is an average over 20 runs, and the error bars indicate 95% confidence interval.



(a) $r = 0.3R$. (b) $r = 0.5R$. (c) $r = 0.7R$.

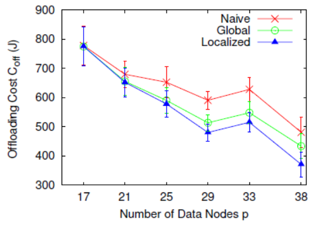
Fig. 6. Performance comparison when $R = 1m$.

Effect of Varying *p*. Fig. 6 shows the offloading cost of three algorithms when *R* = 1*m*. Where *r* is varied from 0.3*R*, 0.5*R*, to 0.7*R*. In each case, the valid range of *p* (using Equation 2) is found and *p* is increased from its smallest valid value to its largest valid value. In each case, it is observed that Global and Localized perform similarly, but both perform better than Naive. This demonstrates that data replication helps reducing offloading cost. It is also observed that the performance improvement of Global and Localized upon Naive gets larger when increasing *p* (it is around 10% at the last *p* value of each case). This is because with more data nodes in the network, the aggregation path gets longer; therefore potentially more storage nodes are on aggregation paths, which works favorably for data replication algorithms. Finally, it is observed that with increase of p, the offloading cost of three algorithms generally decreases. This is because the available storage spaces in the network is $(|V| - p) \times m$. With the increase of p, it has fewer available spaces and less overflow data to offload after aggregation, resulting in less offloading cost.

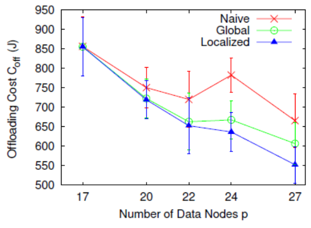
Effect of *R* = 5*m* and *R* = 10*m*. Next, we compare the algorithms when $R = 5m$ and $R = 10m$. Fig. 7 and 8 clearly show that Localized performs better than Global, which performs better than Naive.

In most cases, performance improvement of Localized upon Naive is around 20%. With the increase of the ratio R/m, both the range and the number of valid p decrease (according to Equation2), therefore there are fewer number of storage nodes on each aggregation path. For Global, after running the minimum cost-flow algorithm (line 1 of Algorithm 2), most of the storage nodes on the aggregation paths are filled with offloaded data, leaving not much space for data replication. For Localized, however, it always replicates based upon the calculated demand number of each storage node. A few cases were noted wherein the offloading cost increases with the increase of p (such as $p = 33$ in Fig. 7 (a)), it might be because in these a few cases, there are more data nodes that are not involved in aggregation, and each of these data nodes contain R amount of overflow data, increasing the offloading cost.

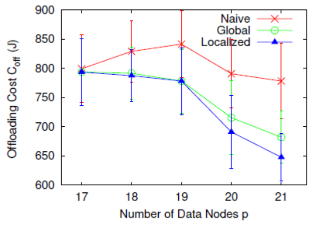
Effect of Varying *r*. Fig. 9 shows the performance comparison of the three algorithms by varying *r*. *R* = 5*m* and *p* = 20 were chosen since they are located in the middle of the simulated parameters.



(a) $r = 0.3R$.

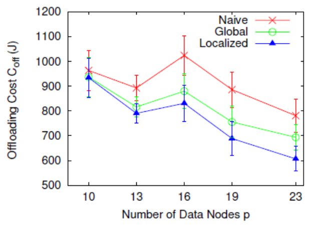


(b) $r = 0.5R$.

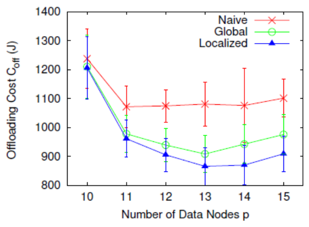


(c) $r = 0.7R$.

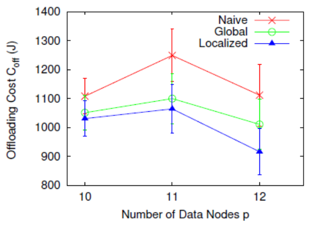
Fig. 7. Performance comparison when $R = 5m$.



(a) $r = 0.3R$.



(b) $r = 0.5R$.



(c) $r = 0.7R$.

Fig. 8. Performance comparison when $R = 10m$.

It shows that if others fixed, with the increase of *r*, the offloading cost of all three algorithms generally increase. This can be explained by Equation 1: with the increase of *r*, *q* increases as well as the *r* amount of overflow data at each aggregator after aggregation. Therefore, the offloading cost increases. However, Fig. 9 also shows that for both Global and Localized, when *r* is increased from 0.5 to 0.7, the offloading cost slightly decreases. This can possibly be attributed to the effect of data replication, since more aggregators are visited, the aggregation paths get longer with the increase of *r*. Therefore, there are potentially more storage nodes on aggregation paths, resulting in less offloading cost due to data replication.

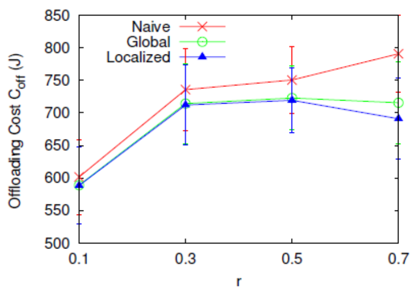
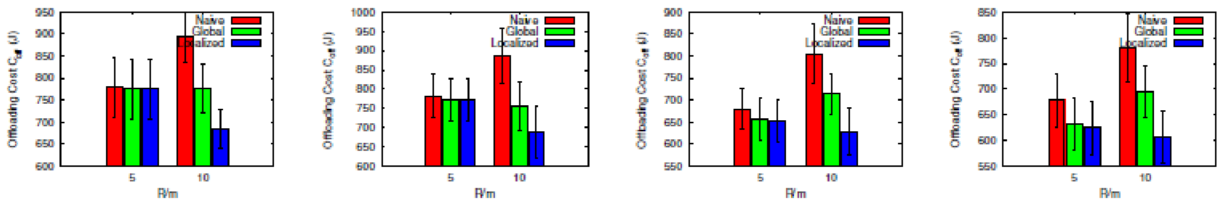


Fig. 9. Performance comparison by varying r. Here, $R = 5m$ and $p = 20$.

Effect of Varying *R/m*. Fig. 10 investigates the effect of *R/m*. Since only *R/m* = 5 and *R/m* = 10 have a common valid range of p, $17 \leq p \leq 23$, when $r = 0.3R$, this set of parameters is adopted. The performance of *R/m* = 5 and *R/m* = 10 is compared by varying p from 17, 19, 21, to 23. First, it is observed that the offloading cost for Naive is always higher in *R/m* = 10 than *R/m* = 5. *R* is doubled from *R/m* = 5 to *R/m* = 10. However, since the total available storage, being $(|V| - p) \times m$ is fixed, the same amount of overflow data after aggregation are offloaded. Therefore, more overflow data needs to be reduced, resulting in increased q. Therefore the data offloading cost for Naive increases. However, when increasing from $R = 5m$ to $R = 10m$, the offloading cost for Localized always decreases, while it could be either way for Global. This again demonstrates the effectiveness of this replication algorithm, with increased q, longer aggregation paths exist, which allows more storage nodes to store replicated data.



(a) $p = 17$. (b) $p = 19$. (c) $p = 21$. (d) $p = 23$.

Fig. 10. Performance comparison by varying R/m. Here, $r = 0.3$.

Chapter 7

CONCLUSION AND FUTURE WORK

In this thesis, two categories of solutions to solve overall storage overflow problem in sensor networks were designed. Proposed first was a naive two-stage solution DAO-N, wherein data offloading strictly follows data aggregation. Proposed next was DAO-R: an integrated method based upon data replication techniques, in order to further reduce energy consumption. The data replication algorithms used integrate data aggregation and data offloading to achieve more energy efficiency. A sufficient condition to solve DAO-R optimally was also given. Using extensive simulations, it was shown that DAO-R outperforms DAO-N by around 20% in terms of energy consumption under different network parameters. As future work, consideration will be made in that different data nodes could have different amounts of overflow data, as well as different storage nodes could have different storage capacity. More dynamic scenarios will also be considered: for example, some nodes could deplete their battery power.

REFERENCES

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

[2] Costas Busch, Malik Magdon-Ismail, Fikret Sivrikaya, and Bulent Yener. Contention-free mac protocols for wireless sensor networks. In *Proc. of DISC*, pages 245–259, 2004.

[3] R. Cristescu, B. Beferull-Lozano, M. Vetterli, and R. Wattenhofer. Network correlated data gathering with explicit communication: Npcompleteness and algorithms. *IEEE/ACM Transactions on Networking*, 14:41–54, 2006.

[4] Deepak Ganesan, Ben Greenstein, Denis Perelyubskiy, Deborah Estrin, and John Heidemann. An evaluation of multi-resolution storage for sensor networks. In *Proc. the 1st international conference on Embedded networked sensor systems (SenSys)*, 2003.

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

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

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

[8] Xiang Hou, Zane Sumpter, Lucas Burson, Xinyu Xue, and Bin Tang. Maximizing data preservation in intermittently connected sensor networks. In *Proc. of MASS 2012,* pages 448–452.

[9] S. Jain, R. Shah, W. Brunette, G. Borriello, and S. Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *MONET*, 11(3):327–339, 2006.

[10] Jaein Jeong, Xiaofan Jiang, and D. Culler. Design and analysis of microsolar power systems for wireless sensor networks. In *Proc. of INSS 2008*, pages 181 – 188.

[11] Apoorva Jindal and Konstantinos Psounis. Modeling spatially correlated data in sensor networks. *ACM Tran. on Sensor Net*, 2:466–499, 2006.

[12] Milica Stojanovic John Heidemann and Michele Zorzi. Underwater sensor networks: applications, advances and challenges. *Phil. Trans. R. Soc. A*, 370:158 – 175, 2012.

[13] Tung-Wei Kuo and Ming-Jer Tsai. On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: Np-completeness and approximation algorithms. In *Proc. of INFOCOM 2012*, pages 2591 – 2595.

[14] Jian Li, Amol Deshpande, and Samir Khuller. On computing compression trees for data collection in wireless sensor networks. In *Proc. of INFOCOM 2010*, pages 2115–2123.

[15] D. Luo, X. Zhu, X. Wu, and G. Chen. Maximizing lifetime for the shortest path aggregation tree in wireless sensor networks. In *Proc. of INFOCOM 2011*, pages 1566 – 1574.

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

[17] Ioannis Mathioudakis, Neil M. White, and Nick R. Harris. Wireless sensor networks: Applications utilizing satellite links. In *Proc. of PIMRC 2007*.

[18] D. Mosse and G. Gadola. Controlling wind harvesting with wireless sensor networks. In *Proc. of IGCC 2012*.

[19] Luca Mottola. Programming storage-centric sensor networks with squirrel. In *Proc. of IPSN 2010*, pages 1–12.

[20] Bin Tang, Neeraj Jaggi, Haijie Wu, and Rohini Kurkal. Energy efficient data redistribution in sensor networks. *ACM Transactions on Sensor Networks*, 9(2), May 2013.

[21] Bin Tang and Yan Ma. Data resilience via data aggregation: Overcoming overall storage overflow in sensor networks. In *Submitted to SECON 2015*.

[22] 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*.

[23] Y. Wu, S. Fahmy, and N. B. Shroff. On the construction of a maximumlifetime data gathering tree in sensor networks: Np-completeness and approximation algorithms. In *Proc. of INFOCOM 2008*.

[24] X. Xu, M. Li, X. Mao, S. Tang, and S. Wang. A delay-efficient algorithm for data aggregation in multihop wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22:163 – 175, 2011.

[25] Xinyu Xue, Xiang Hou, Bin Tang, and Rajiv Bagai. Data preservation in intermittently connected sensor networks with data priorities. In *Proc. of SECON 2013*, pages 65–73.