Power-efficient Distributed Computing and Data Processing in Wireless Sensor Networks

by

Xi Xu
B.S. Jilin University, China, 2009

Thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Chicago, 2015

Chicago, Illinois

Defense Committee:
Rashid Ansari, Co-Chair and Advisor
Milos Zefran
Ajay Kshemkalyani, Computer Science
Hulya Seferoglu
Ashfaq Khokhar, Co-Chair and Advisor, Illinois Institute of Technology

Copyright by

Xi Xu

2015

TABLE OF CONTENTS

CHAPTER		PAGE
1	**INTRODUCTION**	1
1.1	Introduction	1
1.2	Dissertation Organization and Contributions	2
2	**PARALLEL NONUNIFORM DISCRETE FOURIER TRANSFORM (P-NDFT) OVER RANDOM WIRELESS SENSOR NETWORKS**	7
2.1	Introduction	7
2.2	Preliminaries	10
2.2.1	Nonequispaced Discrete Fourier Transform	10
2.3	Parallel NDFT (P-NDFT)	13
2.3.1	Network Architecture	14
2.3.2	Local Interpolation Step of NDFT algorithm	14
2.3.3	Global FFT Implementation	15
2.3.3.1	Prerequisite Step 1: First Stage of Decimation In Frequency (DIF) FFT	16
2.3.3.2	Prerequisite Step 2: Data Reorganization For Decimation In Time (DIT) FFT	18
2.3.3.3	Proposed Power Efficient FFT (PE-FFT)	19
2.4	P-NDFT Analysis	22
2.5	Performance Appraisement	30
2.5.1	Simulation Settings	30
2.5.2	The Accuracy of Fourier Components	32
2.5.3	Energy Consumption	33
3	**POWER-EFFICIENT HIERARCHICAL DATA AGGREGATION USING COMPRESSIVE SENSING IN WSNS**	39
3.1	Introduction	39
3.2	Related Work	43
3.3	Preliminaries	45
3.3.1	Compressive Sensing	45
3.4	Proposed Data Aggregation Architecture	47
3.4.1	Model and Aggregation Process	47
3.5	Parameters Analysis	53
3.5.1	The Amount of Data That Needs To Be Transmitted At One Level	53
3.5.2	The Total Amount of Data Transmitted	55

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

3.5.3 Data Compression Ratio . . . 57
3.5.4 Energy Consumption Model . . . 59
3.5.4.1 Single Node Energy Consumption Model . . . 59
3.5.4.2 Data Processing Cost Analysis . . . 60
3.5.4.3 Communication Cost Analysis . . . 61
3.5.4.4 Total Energy in the Hierarchy . . . 65
3.5.4.5 Energy Model Applied in the Hardware . . . 68
3.6 Performance Evaluations . . . 70
3.6.1 Data Field . . . 70
3.6.2 Signal Sparse Basis . . . 70
3.6.3 Sensing Matrix . . . 71
3.6.4 CS Recovery Algorithm . . . 72
3.6.5 Simulation Settings . . . 74
3.6.6 Simulation Results . . . 76
3.6.6.1 Signal Recovery Performance . . . 76
3.6.6.2 Energy Consumption . . . 79
3.6.6.3 Other Related Issues Discussion . . . 80

4 POWER-EFFICIENT NONUNIFORM 2-D FOURIER ANALYSIS USING COMPRESSIVE SENSING IN WSNS . . . 82
4.1 Introduction . . . 82
4.2 Related Work . . . 84
4.2.1 NDFT algorithm . . . 84
4.2.2 Global Separable 2D FFT Formulation . . . 84
4.3 Proposed Power-efficient NDFT Implementation Design . . . 85
4.3.1 Initial Data Reduction Operation . . . 86
4.3.2 Initial Data Shuffling Operation . . . 86
4.3.3 Global Computation Operation . . . 87
4.4 Comparison of Related Work and Current Work . . . 90
4.4.1 Theoretical Analysis . . . 90
4.4.1.1 Data transmitted . . . 90
4.4.1.2 Energy consumption . . . 91
4.4.2 Performance Evaluation . . . 93

5 ADAPTIVE HIERARCHICAL DATA AGGREGATION USING COMPRESSIVE SENSING (A-HDACS) FOR NON-SMOOTH DATA FIELD . . . 98
5.1 Introduction . . . 98
5.2 Proposed Adaptive HDACS (A-HDACS) Scheme . . . 100
5.3 Analysis of Data Field Sparsity . . . 102
5.4 Performance Evaluation . . . 104
5.4.1 Simulation Settings . . . 104

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

5.4.2 The Nodes Distribution 106
5.4.3 Data Recovery Fidelity 108
5.4.4 Energy Consumption 112

6 SPATIO-TEMPORAL HIERARCHICAL DATA AGGREGATION USING COMPRESSIVE SENSING (ST-HDACS) 113
6.1 Introduction 113
6.2 Spatio-Temporal Hierarchical Data Aggregation using Compressive Sensing 115
6.2.1 Problem Formulation 115
6.2.1.1 Single-instant Data Collection 115
6.2.1.2 Data Collections Over A Period 116
6.2.2 Data Recovery 119
6.2.2.1 Instantaneous Compressive Sensing Recovery 119
6.2.2.2 Latent Matrix Completion Recovery 120
6.3 Performance Evaluation 121
6.3.1 Data Field 121
6.3.2 Simulation Settings 122
6.3.3 Simulation Results 124
6.3.3.1 Data Collection Fidelity 124
6.3.3.2 Energy Consumption 127

7 CONCLUSION AND FUTURE WORK 129
7.1 Conclusion 129
7.2 Future Work 131

CITED LITERATURE 133

VITA 142

LIST OF TABLES

TABLE		PAGE
I	SIMULATION PARAMETERS	31
II	GLOBAL PARAMETERS DEFINITION	47
III	CLUSTER-SPECIFIED && PARAMETERS DEFINITION	48
IV	LEVEL-SPECIFIED PARAMETERS DEFINITION	48
V	PARAMETERS DEFINITION	100

LIST OF FIGURES

FIGURE		PAGE
1	An example of 32-point DIF FFT for the first 16 outputs at the first stage, where the red dash line shows the data communication and black dash line defines the area where the data owned by one sensor	17
2	Data reorganization follows 16 points structure in Figure 1(b), where the red dash line shows the data communication	19
3	An example of 16 points FFT with different implementing structures: where solid black lines represent butterfly communication and red dash lines show data shuffling.	22
4	An example of the network hierarchy evolution from local interpolation step of NDFT algorithm to global FFT implementation with 500 sensors	35
5	SNR	36
6	Histogram of energy consumption for three FFT implementation schemes	37
7	Total energy consumption for all the nodes for three FFT implementation schemes	38
8	Example of Different Data Aggregation Mechanisms	44
9	CS Random Projection	46
10	CS Data Aggregation Architecture	49
11	Theoretical results of energy consumptions in terms of communication and computation cost	69
12	Surface temperature data across the Pacific Ocean from 137 E to 95 W, 8N to 8S on April 22, 2014	74
13	Hierarchical Structure for Network Size 400 on SIDnet-SWANS platform	75
14	Signal Recovery Results for Real Datasets	76

LIST OF FIGURES (Continued)

FIGURE **PAGE**

15 Signal Recovery Results for the Synthetic Datasets 77

16 Signal Recovery Results from Each Level 78

17 Transmission Energy Cost Distribution for Different Network Sizes . . . 78

18 Total Energy Consumption for Different Network Size 79

19 Transmission Energy Consumption Comparison 92

20 Simulated Execution Time Comparison 93

21 Distribution of transmission energy consumption comparison for 400 nodes with cluster size 4 95

22 Signal-to-noise ratio results 96

23 Packet collision ratio comparison for 400 nodes 97

24 An example of a smooth data field with fluctuations and its corresponding logical tree in HDACS and A-HDACS 105

25 Data Fields and their corresponding DCT Domain 107

26 The SIDnet simulation results of A-HDACS and HDACS with network size 400: black nodes denote CS-enabled nodes, gray nodes denote CS-disabled nodes, white nodes are the leaf nodes at level one, and red node denotes the sink. 109

27 MSE versus DCT truncation threshold with network size 400 110

28 Data recovery mean square error (MSE) results 111

29 Total Transmission Energy Cost versus Different Network Sizes 112

30 Data Model 116

31 The Sea Surface Temperature Data Field 122

32 Synthetic Data Fields 123

33 Data Collection Fidelity 125

LIST OF FIGURES (Continued)

FIGURE **PAGE**

34 Total Amount of Energy Consumption versus Network Sizes For Different Data Fields 126

35 The Energy Consumption Ratio of Different Choices of $\lambda < 1$ over $\lambda = 1$ for Each Node in a 300-node Networks 127

LIST OF ABBREVIATIONS

WSNs	Wireless Sensor Networks
NDFT	Nonuniform Discrete Fourier Transform
CS	Compressive Sensing
HDACS	Hierarchical Data Aggregation using Compressive Sensing
A-HDACS	Adaptive Hierarchical Data Aggregation using Compressive Sensing
ST-HDACS	Spatial-Temporal Hierarchical Data Aggregation using Compressive Sensing

SUMMARY

The concept of ubiquitous or pervasive computing (1) has exerted a tremendous influence on the development of novel networks that incorporate a new paradigm of interactions between humans and the physical world. Wireless Sensor Networks (WSNs) (2) (3) enable such interactions that are broadly applicable in many areas including: measurements of physical phenomena (temperature, pressure, humidity, hazardous material); transportation (traffic congestion, bridge structure) ; industrial control, assembly line control and diagnostics; and military applications (task tracking, event detection). The sensors used in WSNs are usually severely constrained in available energy while they may participate in energy-consuming tasks of data communication, aggregation, and interpretation. Power efficiency is one of the key objective that all data gathering and distributed computing tasks aim to achieve.

On the other hand, advances in semiconductor technology enables ever more computing power for sensors. Clever utilization of the computational capability of sensors for in-network processing is essential for reducing the communication cost caused by data transmission redundancy in most WSN applications. It should be noted that compared with communication cost, the cost of the computation is almost negligible. The objective in most applications hence is to balance tradeoffs between computation and communication efficiency and the attendant power dissipation. Recognizing this, our research targets strategies to achieve the valuable information or data of interest over WSNs in a distributed fashion by factoring in the power efficiency.

Towards this end, the focus of this thesis is mainly on the investigation of fundamental distributed computing and data processing tasks in WSNs. In particular we explore power efficient solutions for well-known signal processing tasks such as data gathering and Fourier analysis. We further investigate the adaptive use of compressive sensing to achieve better computation and communication performance in such tasks. We first investigate efficient Fourier analysis of data field based on the randomly distributed sensors in a network and propose to implement Nonuniform DFT (NDFT) algorithm for the data measured from the field. We present a novel structure to realize NDFT implemented on WSNs and also propose an original algorithm with judicious design of computation and communication to reduce the required energy consumption along the data routing path. In our next study, we investigate a power-aware data collection scheme — Hierarchical Data Aggregation using Compressive Sensing (HDACS) for large scale dense wireless sensor networks. We incorporate compressive sensing (CS) in a multi-level data aggregation hierarchy to shrink data volume for transmission and demonstrate how it works more efficiently and effectively when it is implemented in the hierarchical data gathering structure. We also explore the use of CS and HDACS for efficient and distributed/collaborative computing of NDFT in WSNs. In contrast to the existing state of the art, the methods investigated in this dissertation show significant improvement in terms of execution time, transmission power efficiency, SNR and packet collision phenomenon. Most of the existing CS-based data aggregation schemes for WSNs rely on the ideal assumption that data field are globally smooth, thus they fail to work if the data field is non-smooth. We present an adaptive data aggregation scheme referred to as Adaptive Hierarchical Data Aggregation using Compressive Sensing (A-

HDACS) to perform data aggregation in non-smooth multimodal data fields. Finally, for the spatio-temporal data fields, we observe that the existing power-efficient CS-based data aggregation schemes for WSNs either remove data communication redundancy in the routing path or remove the temporal data redundancy by lowering the sampling rate at each sensor. We introduce Spatio-Temporal Hierarchical Data Aggregation scheme using Compressive Sensing (ST-HDACS) to overcome these shortcomings. ST-HDACS simultaneously incorporates the spatial and temporal data redundancies, and formulates the solution by taking advantage of HDACS-based scheme as well as MC.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Wireless sensor networks (2) (3) consisting of a large number of small, low-power, wireless devices with sensing, computation, and communication capabilities offer an array of powerful functionalities to sense the environment, and provide unprecedented opportunities for many scientific disciplines to observe the physical world. Typical applications (e.g., environmental monitoring, surveillance, tracking), usually result in very different network requirements and communication patterns compared to other types of ad hoc network scenarios. The area of communications and protocol design for sensor networks has been widely investigated and a large body of work exists that aims to improve the protocols performance.

The limited availability of battery power in sensors and difficulty that exists in the replenishment of the power resource has spurred research to design power-aware protocols and algorithms for wireless sensor networks. The power consumption in a sensor node is mainly attributed to sensing, communication, and data processing tasks. Numerous studies (4) (5) have shown that transmission cost is significantly higher than computation cost. Developing power-efficient solutions to reduce the required communication cost is therefore highly desirable.

Besides, although some work has been presented from the perspective of power efficient WSN applications mainly dealing with the application layer of the network stack, there are still

many unexplored areas (2) (3). A key area that needs additional attention is the power-efficient implementation of some fundamental tasks which are repeatedly performed within the sensor networks, dealing with data aggregation, data analysis and processing, data implementation and so on, etc. Since power efficiency as a critical index essentially determines the lifetime of the network and has a huge impact on the performance of the application, it is important to study WSN in implementing common processing tasks in a power-efficient manner for a variety of applications and economize their required energy consumption. Based on all these considerations, this dissertation studies power-efficient solutions for key data gathering and analysis tasks and presents power efficient distributed/collaborative implementations of these tasks in WSNs.

1.2 Dissertation Organization and Contributions

Chapter 2 addresses the problem of performing in-network distributed Fourier analysis of non uniformly sensed field. Fourier analysis, an exemplar of distributed data communication and processing task, is widely used in the signal processing field. Past work has presented some energy-efficient in-network Fourier transform computation algorithms devised only for uniformly sampled one-dimensional (1D) sensor data, which may not be directly extendable to 2D uniformly sampled grids. However, sensors are usually randomly distributed over a 2D plane in practice. So the conventional two- dimensional Fast Fourier Transform (2D FFT) defined for data sampled on uniform grids is not directly applicable in such environments. We address this problem by designing a distributed hybrid structure consisting of local Nonequispaced Discrete Fourier Transform (NDFT) and global FFT computation. First, NDFT method is applied in a

suitable choice of clusters to get the initial uniform Fourier coefficients with allowable estimation error bounds. We investigated both classical linear and generalized interpolation methods to compute NDFT coefficients within each cluster. A separable 2D FFT is performed over all these clusters by employing our proposed energy-efficient 1D FFT computation that reduces communication costs using a novel bit index mapping strategy for data exchanges between sensors. The proposed techniques are implemented in a SIDnet-SWANS platform to investigate the communication costs, execution time, and energy consumption. Our results demonstrate reduced execution time and improved energy consumption when compared with existing work.

Recent research on energy-efficient data gathering in WSNs has explored the use of the Compressive Sensing (CS) to parsimoniously represent the data. However, the performance of CS-based data gathering methods has been limited since the approaches failed to take advantage of judicious network configurations and effective CS-based data aggregation procedures. In chapter 3, a novel Hierarchical Data Aggregation method using Compressive Sensing (HDACS) is presented, which combines a hierarchical network configuration with CS. Our key idea is to set multiple compression thresholds adaptively based on cluster sizes at different levels of the data aggregation tree to optimize the amount of data transmitted. The advantages of the proposed model in terms of the total amount of data transmitted and data compression ratio are analytically verified. Moreover, we formulate a new energy model by factoring in both processor and radio energy consumption into the cost, especially the computation cost incurred in relatively complex algorithms. We also show that communication cost remains dominant in data aggregation in the practical applications of large-scale networks. We use both

the real-world data and synthetic datasets to test CS-based data aggregation schemes on the SIDnet-SWANS simulation platform. The simulation results demonstrate that the proposed HDACS model guarantees accurate signal recovery performance. It also provides substantial energy savings compared with existing methods.

Chapter 4 readdresses the problem of distributed computing of NDFT algorithm on WSNs this time using compressive sensing. We leverage compressive sensing (CS) to reduce the amount of data and global communication, thus allowing the use of a few random measurements to adequately represent sparse signal. Our main idea is to organize 2D random deployment of sensors into a hierarchy of clusters. A local interpolation step is performed in the clusters at the lowest level to convert a nonuniform grid into a uniform grid. The global 2D Fast Fourier Transform (FFT) is then implemented using a multi-resolution data aggregation architecture and exploiting CS to reduce data transmission. Theoretical analysis as well as SIDnet-SWANS based simulations demonstrate significant advantages of the proposed method over existing state of the art, in terms of execution time, transmission energy efficiency, signal-to-noise ratio and communication overhead.

Previous work in WSN has used CS under the assumption that data field is smooth with negligible white Gaussian noise. In these schemes signal sparsity is estimated globally based on the entire data field, which is then used to determine the CS parameters. In more realistic scenarios, where data field may have regional fluctuations or it is piecewise smooth, existing CS based data aggregation schemes yield poor compression efficiency. In order to take full advantage of CS in WSNs, chapter 5 proposes an adaptive aggregation scheme referred to as

Adaptive Hierarchical Data Aggregation using Compressive Sensing (A- HDACS). The proposed schemes dynamically determines sparsity values based on signal variations in local regions. We prove that A-HDACS enables more sensor nodes to employ CS compared to the schemes that do not adapt to the changing field. Also, the simulation results demonstrate improvement in energy efficiency and accuracy in signal recovery.

In chapter 6, we observe that existing CS-based data aggregation methods can be categorized as either those that apply CS spatially along the routing path to minimize the amount of data to be communicated from multiple sensors, or those that seek to minimize the amount of data by applying CS temporally at each sensor. A recently reported scheme that is described as a Spatio-Temporal CS scheme randomly selects a subset of data but does not apply compression in the routing path. In this chapter, we formulate a spatio-temporal data collection model in WSNs and refer to it as Spatio-Temporal Hierarchical Data Aggregation using Compressive Sensing (ST-HDACS).In the spatial domain, a multi-level hierarchy of several clusters is formulated. At different levels of the data aggregation tree, multiple compression thresholds are applied adaptively based on cluster sizes to compress the spatial data to be communicated in the routing path, aiming at removing spatial data transmission redundancy. Additionally, for each time snapshot of data collected in the network, a subset of nodes are randomly selected and designated for data sensing and transmission. After performing data collection over a designated time period, a Matrix Completion (MC) problem is executed to recover the data for the entire network over the full data collection period. As only a subset of nodes participate the data aggregation for each data collection instance, the redundancy inherent in the tem-

poral data transmission is eliminated. The performance of the proposed method is evaluated and it is demonstrated that ST-HDACS scheme reduces the amount of data for transmission and improves the associated energy consumption more effectively than existing CS-based data aggregation schemes.

CHAPTER 2

PARALLEL NONUNIFORM DISCRETE FOURIER TRANSFORM (P-NDFT) OVER RANDOM WIRELESS SENSOR NETWORKS

2.1 Introduction

Fourier analysis is one of the most fundamental and essential data analysis method for extracting key features of data field. Since such operations may be performed repeatedly and power efficiency is crucial for sensor network, it is imperative to economize on the energy dissipated on such tasks. Chiasserini et al. (6) firstly investigated Fourier analysis on WSNs using one-dimensional (1D) Fast Fourier Transform (FFT) implemented on sensors equally placed and introduced a two-sensor redundancy-free butterfly computation pattern to remove computation redundancy. Based on his work, Cani et al. (7) presented a bit permutation function on the binary representation of each sensor in the middle phase of FFT to reduce the total distance for data communication. Although those techniques suggest a way of tackling FFT implementation on WSNs, equispaced sensor placement on a straight line is not realistic in most practical applications. In this chapter, we consider a more practical scenario: a large area is monitored by arbitrary placed sensors and the information of interest is the data measured by the sensors projected in the Fourier space. The effort represents the first attempt to investigate the following problems.

1). How to implement Fourier analysis based on the observations from two-dimensional ran-

domly distributed sensor networks?

2). How to minimize the energy consumption especially in terms of communication cost?

A vast amount of research has been done to investigate Nonuniform DFT (NDFT) (8) (9)(10) (11) as a fast implementation of multivariate DFT for nonequispaced data in the applied and numerical mathematics. However it has never been examined in any application on WSNs. According to the nonequispaced characteristic of data captured by sensors, we believe this study is the first one presented to utilize NDFT algorithm to implement Fourier analysis on WSNs. The key idea behind NDFT is to use interpolation to get an enlarged over-sampled uniform data set from a given nonuniform data set and then compute the regular DFT. Note that for a particular position needed to be interpolated, a limited amount of data nearby involved will be enough to obtain Fourier components. This property motivates us to develop a novel hierarchical approach to realize NDFT algorithm combining local interpolation in a predefined cluster with global uniform DFT among clusters. Since global uniform DFT can be implemented in parallel in a row-wise and column-wise fashion, we call our scheme as Parallel-NDFT (P-NDFT).

In the meantime, considering energy efficiency as the most crucial indicator of sensor networks especially in terms of the long-haul data communication cost, a power-efficient implementation of global uniform DFT computation is essentially important. Whereas some strategies were presented in previous work (6) (7) these methods contribute very little for power saving purpose. We notice that bit permutation function introduced in paper (7) to reduce the data transmission distance only for one stage of FFT. This observation drives us to explore bit per-

ambulation function and we thereby propose a novel Power-efficient FFT (PE-FFT) for the global uniform DFT implementation to optimize the data communication distance at the stage level.

In summary, our contributions are multifold:

1. We formulate general framework for a realistic randomly placed sensor networks and address collaborative in-network Fourier analysis on sensor networks.
2. Our work is the first effort to employ NDFT algorithm to implement Fourier analysis in two-dimensional random distributed sensor networks with focus on communication.
3. We designed a original Parallel NDFT (P-NDFT) scheme, a hybrid structure consisting of local interpolation step within a predefined cluster and global Fast Fourier Transform (FFT), to implement NDFT algorithm in a distributed fashion.
4. In terms of global Fast Fourier Transform steps, based on the conventional FFT structure and previous work in paper (7), we propose an effective and novel algorithm working on the binary index of each sensors. With a little extra data shuffling cost, we essentially reduce the required communication distance as well as its associated energy consumption. We refer to this scheme as Power-efficient FFT (PE-FFT).

2.2 Preliminaries

2.2.1 Nonequispaced Discrete Fourier Transform

For a d dimension space, we define $\prod^d := [-\frac{1}{2}, \frac{1}{2})^d$, $I_N := \{(N \prod \bigcap \mathbb{Z})^d\}$. For $x_k \in \prod^d, v_j \in (N \prod)^d$ and $f_k \in \mathbb{C}$, a generalized discrete Fourier transform for a collection of nonequispaced data (NDFT) is expressed as (8)(12)(13):

$$f(v_j) = \sum_{k \in I_N} f_k e^{-2\pi i x_k v_j} \ (j \in I_M) \tag{2.2.1}$$

For arbitrary nodes, the direct evaluations complexity is $O(N^d M^d)$. If data is sampled uniformly in both time and frequency space, i.e. $x_{\mathbf{k}} := \frac{k}{N}(k \in I_N)$ and $v_j := j(j \in I_N)$, the well-known Fast Fourier transform (FFT) can be employed to speed up the computation of $f(v_j)$ and it reduces the arithmetical operations to $\mathrm{O}(N^d \log N)$.

In this work, our major interest focuses on how to obtain uniform frequency components from non-uniform time or spatial data, i.e. we need to find an effective way to implement the formula:

$$h(k) := \sum_{j \in I_N} f_j e^{-2\pi i k v_j / N} \ (k \in I_M) \tag{2.2.2}$$

Among several research results (9) (10) (11), various algorithms have been investigated in paper (8) to compute formula (Equation 2.2.1), Let us first start with gridding algorithm.

Let $g(x) := \sum_{j \in I_N} f_j \varphi(x + \omega_j)$, where φ is a one-periodic function with uniformly convergent Fourier series. By taking Fourier coefficients on both sides, we obtain:

$$\begin{aligned} c_k(g) &= \int_{\prod^d} \sum_{j \in I_N} f_j \varphi(x + \omega_j) e^{2\pi i k x} dx \\ &= \sum_{j \in I_N} f_j e^{-2\pi i k w_j} \int_{\prod^d} \varphi(x + \omega_j) e^{2\pi i k (x + \omega_j)} dx \\ &= \sum_{j \in I_N} f_j e^{-2\pi i k w_j} c_k(\varphi) \\ &= h(k) c_k(\varphi) \end{aligned}$$

Where $c_k(\varphi) = \int_{\prod^d} \varphi(x) e^{2\pi i k x} dx$. Therefore, $h(k)$ can be obtained from $c_k(g)$ and $c_k(\varphi)$. For computing $c_k(g)$,

$$\begin{aligned} c_k(g) &= \int_{\prod^d} \sum_{j \in I_N} f_j \varphi(x + \omega_j) e^{2\pi i k x} dx \\ &= \frac{1}{n^d} \sum_{j \in I_N} f_j \int_{l \in \{(n \prod)^d\}} \varphi(\omega_j - \frac{l}{n}) e^{-2\pi i k l / n} dl \\ &\approx \frac{1}{n^d} \sum_{j \in I_N} \sum_{l \in I_n} f_j \varphi(\omega_j - \frac{l}{n}) e^{-2\pi i k l / n} \end{aligned}$$

In this formulation, an *aliasing error* is introduced from the step of applying discrete summation to approximate the integral. Furthermore, we replace φ by its truncated version ψ, which introduces a *truncation error*. In summary, we approximate the Fourier coefficients of f as:

$$h(k) = \frac{1}{n^d c_k(\varphi)} \sum_{j \in I_N} \sum_{l = \lfloor \omega_j n \rfloor + m}^{\lceil \omega_j n \rceil - m} f_j \psi(\omega_j - \frac{l}{n}) e^{-2\pi i k l / n} \tag{2.2.3}$$

where $m \in \mathbb{Z}$ and it determines the truncation size. Comparing equation (Equation 2.2.2) with equation (Equation 2.2.3), we get:

$$e^{-2\pi i k v_j/N} \approx \frac{1}{n^d c_k(\varphi)} \sum_{l=\lfloor \omega_j n \rfloor + m}^{\lceil \omega_j n \rceil - m} \psi(\omega_j - \frac{l}{n}) e^{-2\pi i k l/n} \tag{2.2.4}$$

The basic idea behind NDFT algorithm is that n equispaced samples, where $n > N$ are first obtained from N nonequispaced samples, and then uniform d-variate $N-$point Discrete Fourier Transform (DFT) is computed from n equispaced data. Lastly, the outcome $c_k(g)$ is scaled by the Fourier coefficients of the window function. The detailed steps are shown in Algorithm 1.

Algorithm 1 NDFT algorithm

Input: $N \in \mathbb{N}, \delta > 1, n := \delta N, w_j \in \prod^d, f_k \in \mathbb{C}\ (j, k \in I_N)$. Define $J_{m,n}(l) := \{j \in I_N : l - m \leq n w_j \leq l + m\}\ (m \in \mathbb{Z})$.

Step 1. Choose suitable window function $\psi(.)$. Precompute its Fourier coefficients $c_k(\psi)\ (k \in I_N)$.

Step 2. Set

$$\tilde{g}_l := \sum_{j \in J_{n,m}(l)} f_j \psi(w_j - \frac{l}{n})\ (l \in I_n)$$

Step 3. Compute by d-variate FFT

$$\tilde{c}_k(g) := n^{-d} \sum_{l \in I_n} \tilde{g}_l e^{-2\pi i k l/n}\ (k \in I_N)$$

Step 4. Compute

$$\tilde{h}(k) := \frac{c_k(g)}{c_k(\psi)}\ (k \in I_N)$$

.

Output: $\tilde{h}(k)$ is the approximate Fourier coefficients of $h(k)$.

Notice that step 2 of NDFT algorithm can be interpreted as an interpolation process, where the application of a truncated window function localizes the associated given points for the interpolated positions. This fact leads us to perform it in a local fashion so as to avoid the long distance communication in WSNs. Besides, step 3 involves Discrete Fourier Transform (DFT) for uniformly sampled data, which drives us to utilize FFT to speed up the computation speed and in the meantime reduce the related data communication across the clusters. Based on these observations, we design a novel distributed NDFT implementation scheme to realize NDFT algorithm in WSNs, which is presented in the following sections.

2.3 Parallel NDFT (P-NDFT)

Based on the facts that 1). Our major attention is focused on improving power efficiency especially for global implementation in terms of long distance communication, which consumes the largest percentage of battery energy; 2). global implementation with respect to 2D FFT is implemented in parallel in a row-wise and a column-wise 1D FFT separately, we refer to this scheme as Parallel NDFT, abbreviated as P-NDFT, to implement NDFT algorithm in WSNs. P-NDFT is designed on the basis of a novel hybrid structure consisting of local processing and global fast Fourier implementation. Additionally, an original power-efficient FFT (PE-FFT) structure is devised in the global FFT implementation step, which aims to shorten the total global communication distance so as to reduce the related power consumption. We prove that PE-FFT outperforms the traditional and other FFT configurations with the same target of power efficiency shown in paper (7).

2.3.1 Network Architecture

In the local processing of P-NDFT, network architecture consists of one-layer clusters, which are configured in a way that nodes within a pre-defined square space belong to one cluster. The identical square space for each cluster ensures the whole coverage of the entire data field and no overlap between any two clusters. In addition, each node within a cluster is assumed to be able to communicate with each other directly, that is, all the nodes in the same cluster are within a one-hop communication distance. Cluster heads can exchange information with cluster heads in other clusters and its slave nodes talk to the other nodes beyond its cluster via the cluster head.

During the global implementation, in order to balance the computation and communication cost and to make sufficient usage of network computation capability, we define *tile*, a direct division of a cluster with finer area, targeting at distributing the computation task among nodes as many as possible. Each *tile* is required to include at least one sensor and we call it as *FFT node*. *FFT node* will participate data communication in the global area.

In order to satisfy network architecture requirement and computation of $N*N$ points Fourier coefficients, we need to place at least one node within *tile*, which is $1/N^2$ of the whole field. This constraint can be realized when the sensors are deployed in the first place.

2.3.2 Local Interpolation Step of NDFT algorithm

Since an interpolation step in the NDFT algorithm involves a small amount of data located in a limited area, it leads us to perform this step within a cluster. In the local processing, cluster head takes charge in collecting data from its own descendent nodes. It is important to

note that data from neighboring clusters is also needed in order to improve the interpolation accuracy and especially for the interpolated points close to the boundary of two cluster heads.

The interpolation step of NDFT algorithm expands the number of the data and maps them to a uniform grid from a given smaller set of nonuniform data, in which a suitable choice of window function plays a critical role in the control of associated errors.

The choice of window function: As we mentioned above, NDFT algorithm produces *aliasing error* and *truncation error*. In order to keep them small, a periodic window function φ should be well localized in time/spatial domain and frequency domain respectively. Several window functions, such as *Gaussian window*, *B-spline window*, *Sinc window*, *Kaiser-Bessel window* are discussed and shown to have their *aliasing error* and *truncation error* in an acceptable range under certain circumstances. In P-NDFT, we choose the Gaussian function as the most fundamental and simple window function with desirable accuracy, as shown in paper (12) (14).

2.3.3 Global FFT Implementation

After uniform data are obtained from the local processing step, the problem has been turned into the computation of a regular 2D DFT. Since 2D DFT can be implemented as a separable row-wise and column-wise 1D FFT, we will only consider 1D FFT problem: n data derived from interpolation step of NDFT algorithm to compute N-point FFT. In the following implementation, we only consider the scenario when the length of a sequence of data for the DFT computation is a power of 2 and $\alpha = 2$, i.e. $n = 2N$.

Note that the numbers of the input and output data samples are different, FFT cannot be directly utilized as the butterfly structure requires both data with the same sequence length.

Besides, we should consider how to retrieve Fourier coefficients when the entire distributed computation task is finished. It is desirable that users can get the output in a sequential order without struggling to figure the order of Fourier coefficients. Based on these considerations, we propose a hybrid structure to tackle this problem by combining the first stage of Decimation In Frequency (DIF) FFT with the entire Decimation In Time (DIT) FFT.

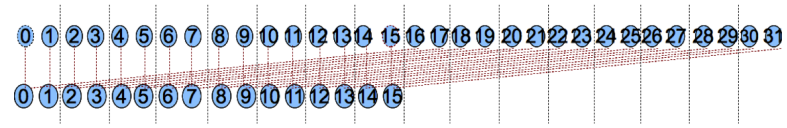
2.3.3.1 Prerequisite Step 1: First Stage of Decimation In Frequency (DIF) FFT

One property of DIF FFT is that after the first stage of the DFT computation, one half of outputs contribute to computing the even components and the other half are for computing the odd components of the Fourier coefficients, which are reflected in Formula (Equation 2.3.1) and (Equation 2.3.2) . Here $W_N^{tl} = e^{-2\pi tl/N}$, $W_n^t = e^{-2\pi t/n}$ and $n = 2N$:

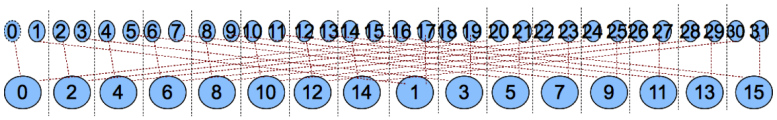
$$X[2l] = \sum_{t=0}^{N-1} (x[t] + x[N+t]) W_N^{tl} \quad (0 \leq l \leq N-1) \tag{2.3.1}$$

$$X[2l+1] = \sum_{t=0}^{N-1} (x[t] - x[N+t]) W_n^t W_N^{tl} \quad (0 \leq l \leq N-1) \tag{2.3.2}$$

We observe that the even and odd frequency components in the following stages of DIF FFT can be independently computed. It is feasible to reduce the data size from $n = 2N$ to N and retrieve the partial Fourier coefficients from the original data. Suppose we want to get N even frequency components of n original data. (Odd frequency components can be obtained in the same way.) Directly applying DIF FFT structure into P-NDFT will narrow down the following computation of FFT in a limited area involving only a small friction of sensors, which



(a) A traditional DIF FFT structure



(b) A proposed data transmission structure

Figure 1. An example of 32-point DIF FFT for the first 16 outputs at the first stage, where the red dash line shows the data communication and black dash line defines the area where the data owned by one sensor

fails to take full advantage of network computation capability. Therefore, we propose a new data transmission structure to implement the first stage of DIF FFT. For n data, samples used in computing the first step of DIF FFT, the newly designed structure transmits the data with odd index l, where $0 \leq l \leq N-1$, to its butterfly partner. This is different from the first stage of regular DIF FFT structure where each data sample with index l, where $0 \leq l \leq N-1$, receives data from its butterfly parter with index m, where $N \leq m \leq n$. Figure 1(a) gives an example of a 32-point traditional butterfly structure for the first 16 outputs at the first stage of DIF FFT, where the red dash line shows the data communication and black dash line defines the data owned by one sensor. The proposed data transmission structure is shown in Figure 1(b). As we can see, one sensor only owns one data, which we refer to them as *FFT nodes* and all *FFT nodes* are evenly spread in the entire field.

In this step, cluster heads can work as proxy to shuffle data for *FFT nodes*. As cluster heads own all the data from the interpolation step of NDFT algorithm and the destination of data transmission is pre-designated, it is easier and more energy-efficient to bundle up all the data that needs to be transmitted from one sensor than multiple sensors, After cluster heads distribute the data to *FFT nodes*, the network hierarchy evolves from clusters into *tile*, in which *FFT nodes* will participate to perform the global DIT FFT step.

2.3.3.2 Prerequisite Step 2: Data Reorganization For Decimation In Time (DIT) FFT

When the network finishes the computation task, it is critical to examine how to build a suitable channel for users to query frequency components outputs in a certain range. An ideal scenario could be one in which each *FFT node* holds one 2D frequency component placed in a sequential order. Recall that each data according to its location in FFT structure can be represented as a binary representation $< b_1 b_2 \dots b_T >$ with each bit in an ascending order, where $T = \log_2 N$ is the total amount of stages for computing N-point DIT FFT. At the final stage, the binary representation will be reversed, i. e. $< b_T \dots b_2 b_1 >$ with each bit in a descending order (15). It means that the input location is in *bit-reversed order* from that of the original input. Therefore, to assure the sequential order of outputs, input data needs to be reorganized in the first place to satisfy *bit-reversed order* condition between input and output Figure 2 shows the data reorganization follows 16 points structure shown in Figure 1(b).

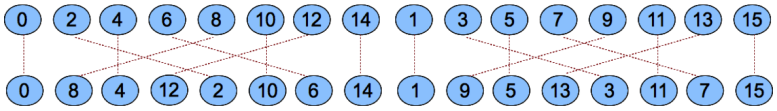


Figure 2. Data reorganization follows 16 points structure in Figure 1(b), where the red dash line shows the data communication

2.3.3.3 Proposed Power Efficient FFT (PE-FFT)

Conventional FFT structure involves a great deal of long distance communication especially when the total amount of stages increases and data transmission consumes the vast majority of energy for a sensor than any other factors. To facilitate the power efficiency of FFT, we propose a Power Efficient FFT (PE-FFT) by reducing the required total transmission cost, or equivalently, reducing the total amount of transmission distance required in FFT.

Two permutation functions are introduced to work on the binary representation of each node at each FFT stage. The basic idea is that by updating binary representation and data shuffling at each stage, PE-FTT brings butterfly partners closer to each other. We know that in the j^{th} stage, where $1 \leq j \leq T$, one complex multiplication is needed if $b_j = 1$, and none if $b_j = 0$. Besides, we also define each *FFT node* with *physical ID* and *logical ID*, where *physical ID* $=< b_1 b_2 \dots b_T >$ is fixed and used to identify physical location of *FFT node* and *logical ID* changes with stage j determined by permutation functions. After prerequisite steps, it starts with initial assignment *logical ID*$_1 =< b_T b_{T-1} \dots b_2 b_1 >$. Let $k = T/2$ represent the middle stage. (If T is odd, k could be either $(T-1)/2$ or $(T+1)/2$).

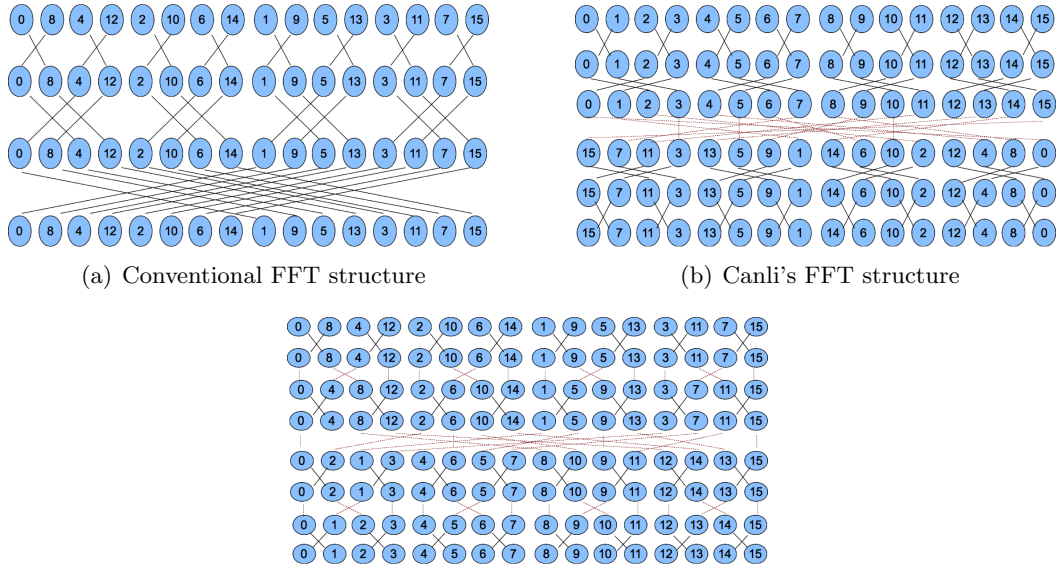
Definition 1. *$Q < . >$ is defined as a right circular shift function performed at stage j, $j \neq k$. If stage $1 \leq j < k$, the leftmost first to $(j+1)^{th}$ bits right circular shift one bit and the other bits are kept unchanged, i.e., if $1 \leq j < k$, $Q < b_{T-j}b_{T-j+1} \ldots b_T b_{T-j-1} \ldots b_{k+1}b_k \ldots b_2 b_1 >=< b_{T-j-1}b_{T-j} \ldots b_T b_{T-j-2} \ldots b_{k+1}b_k \ldots b_2 b_1 >$; on the other hand, if $k < j < T$, the leftmost $(k+1)^{th}$ to $(j+1)^{th}$ bits right circular shift one bit and the other bits are kept unchanged, i.e, if $k < j < T$, $Q < b_1 b_2 \ldots b_k b_{T-(j-k)} \ldots b_{T-1} b_T b_{T-(j+1-k)} \ldots b_{k+2}b_{k+1} >=< b_1 b_2 \ldots b_k b_{T-(j+1-k)} b_{T-(j-k)} \ldots b_{T-1} b_T \ldots b_{k+2} b_{k+1} >$.*

Definition 2. *$P < . >$ is defined as an inverse function performed at the middle stage $j = k$. In $P < . >$, each bit b_j is replaced as b_{T-j}, i.e., $P < b_{k+1} \ldots b_T b_k b_{k-1} \ldots b_2 b_1 >=< b_1 b_2 \ldots b_{k-1} b_k b_T \ldots b_{k+1} >$.*

We design PE-FFT in a way that FFT butterfly communication and computation alternates with data shuffling based on permutation functions with progression of stages. According to DIT FFT, butterfly communication and computation should be performed between two neighboring *FFT nodes* at stage $j = 1$. After this, Q$< . >$ works on the leftmost two bits of *logical ID*, and the binary index representation of a *FFT node* changes from *logical ID*$_1 =< b_T b_{T-1} \ldots b_{k+1} b_k \ldots b_2 b_1 >$ to *logical ID*$_2 =< b_{T-1} b_T \ldots b_{k+1} b_k \ldots b_2 b_1 >$. Data is shuffled among nodes for its newly updated *logical ID* and it proceeds to stage $j = 2$. At stage two, FFT butterfly computation continues and we will show in Proposition 1 that it still take place between the same neighboring node pairs as stage one. And then Q$< . >$ operates the leftmost three bits of its binary representation and it changes from *logical ID*$_2 =< b_{T-1} b_T b_{T_2} \ldots b_{k+1} b_k \ldots b_2 b_1 >$ to *logical ID*$_3 =< b_{T-2} b_{T-1} b_T \ldots b_{k+1} b_k \ldots b_2 b_1 >$. Nodes com-

municate to update their data for their newly designated *logical ID*. This procedure iteratively proceeds until it comes to the middle stage $j = k$, where the binary representation should be *logical* $ID_{k-1} =< b_{k+1}b_{k+2}....b_T b_k b_{k-1} \dots b_2 b_1 >$. After FFT butterfly computation, $P < . >$ works on the binary index and inverses it into *logical* $ID_k =< b_1 b_2 \dots b_{k-1} b_k b_T \dots b_{k+2} b_{k+1} >$. At stage $k < j < T$, PE-FFT proceeds the same way as stage $0 < j < k$ and Q$< . >$ keep playing a key role for data shuffling until the stage reaches to $j = T - 1$, *logical* $ID_{T-1} =< b_1 b_2 \dots b_k b_{k+1} \dots b_{T-1} b_T >$. At the final stage $j = T$, only butterfly communication and computation is needed and we get N Fourier coefficient outputs.

Figure 3 shows an example of 16 points FFT with different implementing structures, where solid black lines represent butterfly communication and red dash lines show data shuffling. Conventional FFT structure depicted in Figure 3(a) shows the distance between two butterfly pairs increases exponentially over the stage. Canli et al. proposed a power-efficient FFT structure in the paper (7), which we call it as Canli-FFT for subsequent reference. Canli-FFT incorporated a new inverse-shuffle complement function for binary representation of each node at the middle stage to reduce related transmission distance. To be specific, at stage $j = k$, if a node with binary index $< b_1 b_2 \dots b_{T-1} b_T >$, inver-shuffle complement function changes it into $< b'_T b'_{T-1} \dots b'_2 b'_1 >$. Figure 3(b) shows a 16-point FFT implementation structure presented in his work. Note that it only optimizes the communication in the middle stage. 16 points FFT implementation structure of PE-FFT algorithm is given in Figure 3(c). We notice that plenty of *FFT nodes* is actually not necessary to shuffle data as $Q < . >$ does not change their *logical*



(a) Conventional FFT structure

(b) Canli's FFT structure

(c) PE-FFT structure

Figure 3. An example of 16 points FFT with different implementing structures: where solid black lines represent butterfly communication and red dash lines show data shuffling.

ID. By adding extra data shuffling cost, Figure 3 visualizes an effective overall transmission distance reduction.

In the following section, we give a theoretical analysis comparing the transmission distance for different FFT implementation schemes.

2.4 P-NDFT Analysis

Proposition 1. *In PE-FFT, binary permutation functions* $Q < . >$ *and* $P < . >$ *guarantee that FFT butterfly computation pairs always are physical neighbors of each other for any stage* j*, where* $0 < j < T$*.*

Proof. At stage j and $0 < j < T$, one butterfly pair is two nodes with only one bit difference at the j^{th} leftmost position and all the other bits the same. At stage one, all butterfly pairs determined by the first leftmost bit b_T are physically neighboring each other. With the progression of stage, $Q < . >$ and $P < . >$ both guarantee that the j^{th} leftmost bit of binary expression is b_T. In the meantime they do not change the values of any bits. Therefore, it proves that $Q < . >$ and $P < . >$ guarantee FFT butterfly computation pairs are always physically neighboring each other at any stage j. □

Proposition 2. *At the final stage T, Fourier coefficients in PE-FFT comes in the same order of the inputs.*

Proof. For a node with binary representation $< b_1 b_2, \ldots b_T >$, after prerequisite step 2, binary index becomes into $< b_T b_{T-1}, \ldots b_1 >$. At each stage $0 < j < T$, $Q < . >$ function and $P < . >$ function shift the order of bits in binary index in a way that it eventually turns out to be $< b_1 b_2, \ldots b_T >$ at the final stage T, which is exactly the same order of the inputs. Therefore, Fourier outputs in PE-FFT comes in a sequential order. □

Notice that transmission cost is closely related to the volume of data communicated and related transmission distance. And during a FFT implementation, each communication only involves a constant unit of data. Therefore, the related transmission distance is the most important indicator for energy efficiency in this work and we claim that the proposed PE-FFT involves the least total communication distance compare with other FFT implementation schemes.

Proposition 3. *Compare with FFT and Canli-FFT, PE-FFT involves the least total data transmission distance.*

Proof. We assume the distance between two neighboring *FFT nodes* as a unit and start to look at 1D case firstly.

1). For PE-FFT, we first look at the distance associated with FFT butterfly computation. Based on Proposition 1, there are $\log N$ stages of butterfly computation and at each stage, where each node transmits one data to its butterfly parter with unit distance. Therefore, we get the distance for butterfly computation over the whole process as: $D_{PE-FFT,b} = N \log N$.

With regard to the distance of data shuffling, symmetric property of PE-FFT structure enables us to look at stage $0 < j < k$. Given an arbitrary node p with *logical ID* $p.logicalID =< b_T b_{T-1} \dots b_2 b_1 >$ and *physical ID* $p.physicalID =< b_1 b_2 \dots b_T >$. Note that at each stage, the transmission distance of data shuffling is the physical distance between a node with *logical ID* at stage j and another node with the same *logical ID* at stage $j+1$. At stage j, $p.logicalID_j =< b_{T-(j-1)} b_{T-(j-2)} \dots b_T b_{T-j} \dots b_2 b_1 >$. Suppose node q is the designated node for exchanging data with node p at stage j. Therefore, we get $p.logicalID_j = q.logicalID_{j+1}$. The *physical ID* of node q is obtained by reversing the leftmost $j+1$ bits of $q.logicalID_{j+1}$ and then reversing the entire binary index, i.e. $q.physicalID =< b_1 b_2 \dots b_{T-j-1} b_{T-(j-1)} b_{T-(j-2)} \dots b_T b_{T-j} >$. We define $D^j_{PE-FFT,p,q}$ as the distance before node p and node q at stage j for data exchanging

operation. Note that the binary indexes of node p and node q have the difference only for the leftmost $T-j-1$ bits.

$$
\begin{aligned}
D^{p,q,j}_{PE-FFT,d} &= |p.physicalID - q.physicalID| \\
&= | < b_1 b_2 \dots b_{T-j-1} b_{T-j} b_{T-j+1} \dots b_T > \\
&- < b_1 b_2 \dots b_{T-j-1} b_{T-(j-1)} b_{T-(j-2)} \\
&\dots \quad b_T b_{T-j} > | \\
&= | < b_{T-j} b_{T-j+1} \dots b_T > \\
&- < b_{T-(j-1)} b_{T-(j-2)} \dots b_T b_{T-j} > | \\
&= |b_{T-j} * 2^j + < b_{T-j+1} \dots b_T > \\
&- (< b_{T-j+1} \dots b_T > * 2 + b_{T-j})| \\
&= |b_{T-j}(2^j - 1) - < b_{T-j+1} \dots b_T > |
\end{aligned}
$$

Hence we get:

$$
D^{p,q,j}_{PE-FFT,d} = \begin{cases} < b_{T-j+1} \dots b_T > & \text{if } b_{T-j} = 0 \\ 2^j - 1 - < b_{T-j+1} \dots b_T > & \text{if } b_{T-j} = 1 \end{cases}
$$

Define Ω as a collection of *FFT nodes* and ω as a collection of stages in FFT. Since node $p \in \Omega$ if $b_{T-j} = 1$, we can always find a node $p' \in \Omega$, with $b_{T-j} = 0$ and the same bits $< b_{T-j+1} \dots b_T >$. Therefore, we obtain:

$$\sum_{p,q \in \Omega} D^{p,q,j}_{PE-FFT,d} = \frac{N}{2}(2^j - 1)$$

And transmission distance over all stages except middle stage is:

$$D^{j \in \omega \setminus \{k\}}_{PE-FFT,d} = 2 \sum_{j=1}^{k-1} \sum_{p,q \in \Omega} D^{p,q,j}_{PE-FFT,d} = \frac{1}{2} N^{1.5} - \frac{1}{2} N \log N$$

For the middle stage $j = k$, *logical ID* of node p at stage k is $p.logicalID_k =< b_{k+1} b_{k+2} \dots b_T b_k$ $b_{k-1} \dots b_2 b_1 >$. Again, if node q is the target node for node p to forward its current data, it satisfies $q.logicalID_{k+1} = p.logicalID_k$. The *physical ID* of node p is obtained by firstly reversing its binary representation $q.logicalID_{k+1}$, then reversing the leftmost k bits and finally reversing the entire binary index again. Therefore, we get $q.physicalID =< b_{k+1} \dots b_T b_1 \dots b_k >$.

Hence, the total transmission distance at stage $j = k$ is:

$$\begin{aligned} D^{p,q,j=k}_{PE-FFT,d} &= |< b_1 b_2 \dots b_k b_{k+1} \dots b_T > \\ &- < b_{k+1} \dots b_T b_1 \dots b_k > | \\ &= |< b_1 b_2 \dots b_k > *2^k + < b_{k+1} \dots b_T > \\ &- (< b_{k+1} \dots b_T > *2^k + < b_1 b_2 \dots b_k >)| \\ &= |(< b_1 b_2 \dots b_k > - < b_{k+1} \dots b_T >) \\ &\quad (2^k - 1)| \end{aligned}$$

Let $t =< b_{k+1} \dots b_T >$ and $t' =< b_1 b_2 \dots b_k >$, and $t, t' \in \{0, 1, 2, \dots 2^k - 1\}$.

$$\begin{aligned} \sum_{p,q \in \Omega} D^{p,q,j=k}_{PE-FFT,d} &= \sum_{p,q \in \Omega} D^{p:t'>t,q,j=k}_{PE-FFT,d} + \sum_{p,q \in \Omega} D^{p:t'<t,q,j=k}_{PE-FFT,d} \\ &= 2 \sum_{t=0}^{2^k-2} \sum_{t'=t+1}^{2^k-1} (t' - t)(2^k - 1) \\ &= \frac{1}{3} 2^k (2^k - 1)(2^{2k} - 1) \\ &= \frac{1}{3} (N^2 - N\sqrt{N} - N + \sqrt{N}) \end{aligned}$$

Hence the transmission distance associated with data shuffling is: $D_{PE-FFT,d} = \frac{1}{3}N^2 + \frac{1}{6}N\sqrt{N} - \frac{1}{3}N + \sqrt{N} - \frac{1}{2}N \log N$

In summary, the total transmission distance in PE-FFT is: $D_{PE-FFT} = D_{PE-FFT,b} + D_{PE-FFT,d} = \frac{1}{3}N^2 + \frac{1}{6}N^{1.5} - \frac{1}{3}N + N^{0.5} + \frac{1}{2}N \log N$.

2). For conventional FFT, the total transmission distance is easily derived as $D_{FFT} = N^2 - N$.

3). For Canli-FFT, paper (7) shows the total transmission distance: $D_{Canli-FFT} = 2N^{1.5} - 2N + IEC(N)$, where $IEC(N)$ is defined as the number of transmissions required to realize the inverse-shuffle complement phase in the middle phase and a precise close expression is not provided. It's easy to conclude that for the transmission distances over all stages except the middle phase $D^{j\in\omega\setminus\{k\}}_{PE-FFT,d} < 2N^{1.5} - 2N$. Following the definition of inverse permutation complement function, for any node $p \in \Omega$ with binary representation $< b_1 b_2 \dots b_{T-1} b_T >$, its data shuffling partner q is the node with binary representation $< b'_T b'_{T-1} \dots b'_2 b'_1 >$. Therefore, we get:

$$IEC(N) = \sum_{p,q\in\Omega} | < b_1 b_2 \dots b_{T-1} b_T > - < b'_T b'_{T-1} \dots b'_2 b'_1 > |$$

For binary subtraction, we use the two's-complement representation of the negative number (15) and let $t =< b_1 b_2 \dots b_{T-1} b_T >$ and $t' =< b_T b_{T-1} \dots b_2 b_1 >$. Therefore,

$$\begin{aligned}
IEC(N) &= \sum_{p,q\in\Omega} | < b_1 b_2 \dots b_{T-1} b_T > - < b'_T b'_{T-1} \dots b'_2 b'_1 > | \\
&= \sum_{p,q\in\Omega} |(b_1 - b'_T) * 2^{T-1} + (b_2 - b'_{T-1}) * 2^{T-2} + \dots + (b_T - b'_1) * 2^0| \\
&= \sum_{p,q\in\Omega} |(b_1 + b_T - 1) * 2^{T-1} + (b_2 + b_{T-1} - 1) * 2^{T-2} + \dots + (b_T + b_1 - 1) * 2^0| \\
&= \sum_{p,q\in\Omega} |(b_1 + b_T) * 2^{T-1} + (b_2 + b_{T-1}) * 2^{T-2} + \dots + (b_T + b_1) * 2^0 - (N-1)|
\end{aligned}$$

We now define a subset $\Omega' \subset \Omega$ where $|\Omega'| = 1/4|\Omega|$ corresponding to $b_1 + b_T = 2$ and another subset $\Omega'' \subset \Omega$ where $|\Omega''| = 1/4|\Omega|$ corresponding to $b_1 + b_T = 0$. The formula above hence can be further expressed as:

$$
\begin{aligned}
IEC(N) &= \sum_{p,q\in\Omega'} |2*2^{T-1} + (b_2 + b_{T-1})*2^{T-2} + \cdots + 2*2^0 - (N-1)| \\
&+ \sum_{p,q\in\Omega''} |0*2^{T-1} + (b_2 + b_{T-1})*2^{T-2} + \cdots + 0*2^0 - (N-1)| \\
&+ \sum_{p,q\in\Omega\setminus\{\Omega',\Omega''\}} |1*2^{T-1} + (b_2 + b_{T-1})*2^{T-2} + \cdots + 2^0 - (N-1)| \\
&\geq \sum_{p,q\in\Omega'} (2^{T-1} + 2^0) + (2^{T-1} + (b_2 + b_{T-1})*2^{T-2} + \cdots + 2^0 - (N-1)) \\
&+ \sum_{p,q\in\Omega''} (N-1) - (2^{T-1} + (b_2 + b_{T-1})*2^{T-2} + \cdots + 2^0)) + (2^{T-1} + 2^0) \\
&= \sum_{p,q\in\{\Omega',\Omega''\}} (2^{T-1} + 2^0) \\
&= N/2(2^{T-1} + 2^0) \\
&= N^2/2 + N/2 \\
&> \sum_{p,q\in\Omega} D^{p,q,j=k}_{PE-FFT,d}
\end{aligned}
$$

This shows that PE-FFT requires a transmission distance that is lower than that of Canli-FFT. For 2D case, comparison result does not changed as all three schemes execute the row-wise and column-wise FFT separately.

In summary, it is clear that PE-FFT outperforms other schemes by reducing the total amount of transmission distance.

□

2.5 Performance Appraisement

2.5.1 Simulation Settings

We evaluated Fourier analysis schemes on SIDnet-SWANS (16), (Simulator and Integrated Development Platform for Sensor Networks Applications) to study their performance. SIDnet is a Java-based visual tool designed to promote run-time interactions with the network and JiST/SWANS (17) (18) (Java in Simulation Time), a discrete-event simulation engine, to simulate all behaviors of a sensor for retrieving real-time parameter readings.

In the simulation, network size M^2 (We use M^2 for dimension transformation convenience, and $M^2, N^2, n^2 \in \mathbb{Z}$) is set up in the range 300 to 800 with an increment of 50 each time. We want to compute $N^2 = 16 * 16$ Fourier coefficients over the whole field from M^2 nonuniform samples measured by sensors, which are first expanded into $n^2 = 32 * 32$ uniformly spaced samples.

In this setting, we establish the relationship for the number of Fourier coefficients, network size and uniformed data size, which satisfies: $N^2 < M^2 < n^2$. The field size is fixed as $4000 * 4000m^2$, and average node distribution density increases from $18.75/\text{km}^2$ to $50/\text{km}^2$. The communication system follows IEEE 802.15.4/4a standards for low data rate wireless personal data networks and data transmission rate is set as 40000bps. Besides, the length of a standard message for transmission is 133 bytes. Theoretically, it takes $133 * 8/40000 = 26.6$ms for one message per hop. We neglect the noise interference from the channel and ignore the possibility of random packet loss. For power consumption, it follows Mica2 Motes specs, where the radio transmission cost is our major concern. The default unit cost of radio transmission is $81\mu J/ms$

TABLE I

SIMULATION PARAMETERS

Parameter	Values
Sensor number	$300 \sim 800$
Area	$4000 * 4000\text{m}^2$
Sensor density	$18.75/\text{km}^2 \sim 50/\text{km}^2$
Radio frequency	2.4GHz
Bandwidth	$4 * 10^4$bps
Antenna gain	4dB
Unit cost of radio transmission	81 $\mu J/ms$

for one message. Table I lists the major parameter assignments in the simulations. Figure 4 shows evolution of the hierarchy of the network consisting of 500 sensors from local processing into global data communication.

The interpolation in the NDFT algorithm is performed using a Gaussian window function given by: $\varphi(x, y) = \frac{1}{\pi b} \exp^{-(\sigma N)^2(x^2+y^2)/b}$, where (x, y) is the coordinate of given data f and $b = \frac{2\sigma}{2\sigma-1}\frac{m}{\pi}$. Notice that NDFT algorithm requires access to data of size N, which is neither practical nor efficient when it comes to the distributed execution. In P-NDFT, each interpolated point is determined by the data within its own cluster. Therefore, the Gaussian function for each cluster in P-NDFT is set up as: $\varphi(x, y) = \frac{1}{\pi b} \exp^{-(\sigma_c N_c)^2(x^2+y^2)/b_c}$, where N_c is the number of data sensed in one cluster, $\sigma_c = n_c/N_c$ and $b_c = \frac{2\sigma_c}{2\sigma_c-1}\frac{m_c}{\pi}$. In the simulation, $8 * 8$ clusters are set up across the whole plane initially. In each cluster, $n_c = 4^2$ points are interpolated uniformly.

2.5.2 The Accuracy of Fourier Components

Ground truth is considered as the scenario in which all the data is accurately collected at sink and the sink implements a centralized NDFT algorithm for the whole data. To examine the output accuracy, we define Signal to Noise Ratio (SNR) as ten times the decimal logarithm of the ratio of P-NDFT outputs to the ground truth. The expression is as follows:

$$SNR_{db} = 10 \log \frac{\sum_{m=0}^{N} ||\boldsymbol{X}_m||_2}{\sum_{m=0}^{N} ||\boldsymbol{X}_m - \hat{\boldsymbol{X}}_m||_2}$$

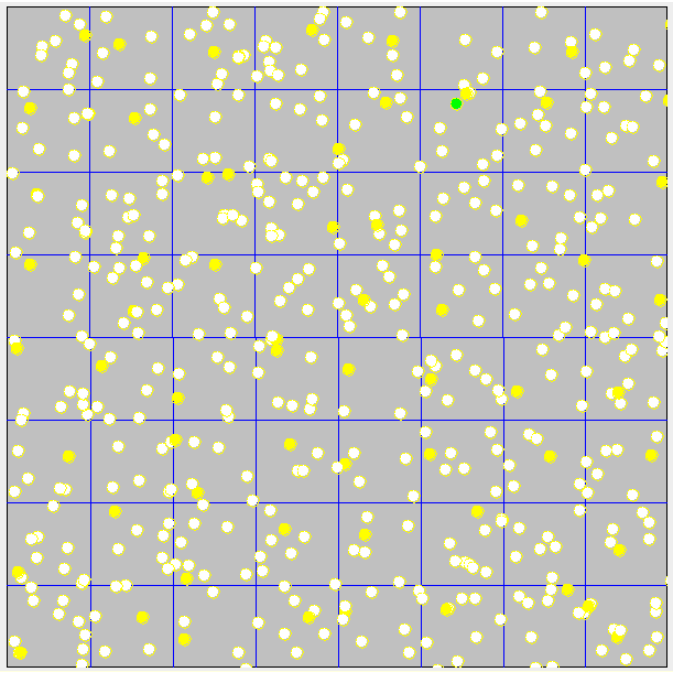
where $\boldsymbol{X_m}$ denotes the m^{th} Fourier coefficients ground truth and $\hat{\boldsymbol{X}}_{\boldsymbol{m}}$ denotes the m^{th} P-NDFT outputs. Figure 5 shows the accuracy of outputs against different network sizes. With the increase in network size, SNR performance improves steadily under different choices of m_c. However, we notice that enlarging the truncation parameter m_c monotonously does not necessarily improve SNR. The reason is that within one cluster, larger m_c raises the value of the variance parameter b of Gaussian function and consequently leads to reduced contribution of each data for interpolation; on the other hand, the available data is limited within a cluster and it affects interpolation accuracy severely. Figure 5 also reflects the fact that the overall level of SNR is low. Data interpolation performed within one cluster compromises the output fidelity. In order to improve output accuracy, the interpolation for a given cluster is performed by including data from its eight closest neighboring clusters. In this scenario, N_c is the number of nodes in a bigger cluster consisting of nine adjacent initial clusters, and n_c expands into 12^2. With a little sacrifice of data transmission redundancy, Figure 5(b) shows the overall SNR

performance has been enhanced remarkably. In addition, we find that $m_c = 0.6\sqrt{n_c}$ gives the highest overall SNR for both cases. Therefore, the right choice of m_c is essential to improving SNR performance.

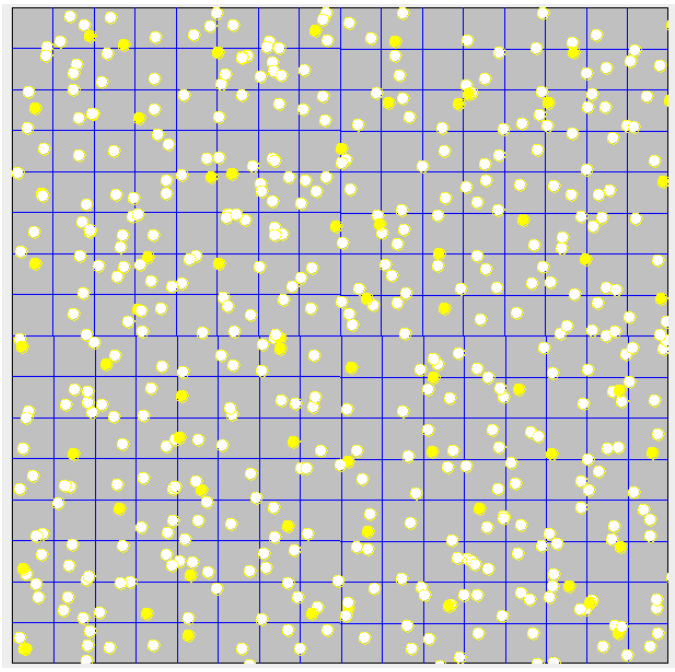
2.5.3 Energy Consumption

For energy consumption, our primary focus is on tasks that require global data communication. For comparison purpose, three FFT implementation schemes: our proposed PE-FFT, conventional FFT, and Canli-FFT are evaluated for P-NDFT to illustrate the advantage of proposed PE-FFT. After the local interpolation processing, all three schemes are executed simultaneously. We plot the histograms of the three FFT schemes in Figure 6 to show the distribution of the energy dissipated per node with two network sizes 400 and 600. In particular, equal communication distance per node at any stage of a conventional makes its histogram smoother than that of other schemes. However, the overall level of energy consumption is much higher than the other schemes. On the other hand, extra data shuffling in PE-FFT scheme and Canli-FFT causes variations of power consumption in different nodes. There is a tradeoff between the homogeneity in power consumption and energy efficiency. However, the objective of this work is to minimize the total communication cost. Figure 6 shows that the overall level of energy consumption of PE-FFT is much lower than other schemes and it validates the effectiveness of PE-FFT in improving the energy efficiency. Moreover, Figure 7 plots the total energy consumption for the whole network against its size for three schemes. Clearly, PE-FFT scheme involves the least amount of energy consumption. In Figure 7, PE-FFT with interpolation performed in one cluster saves 39.73% to 41.84% of the energy dissipated over

Canli-FFT and 50.95% to 57.34% of the energy consumed on the conventional FFT; PE-FFT with interpolation performed including neighboring clusters saves 31.44% to 38.10% of the energy dissipated over Canli-FFT and 44.10% to 53.18% of the energy consumed compared with the conventional FFT.

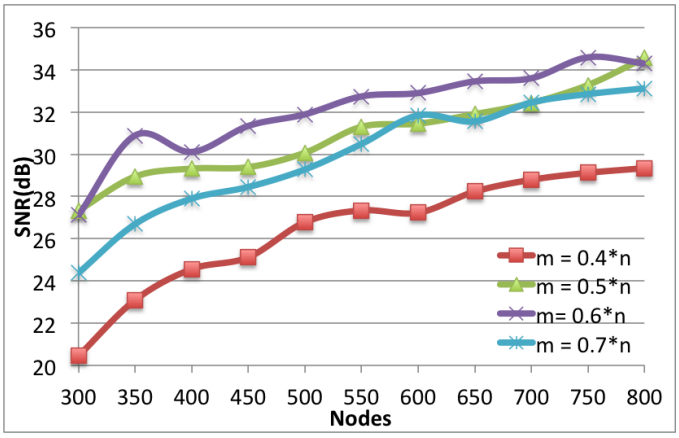


(a) Initial $8 * 8$ clusters for local processing

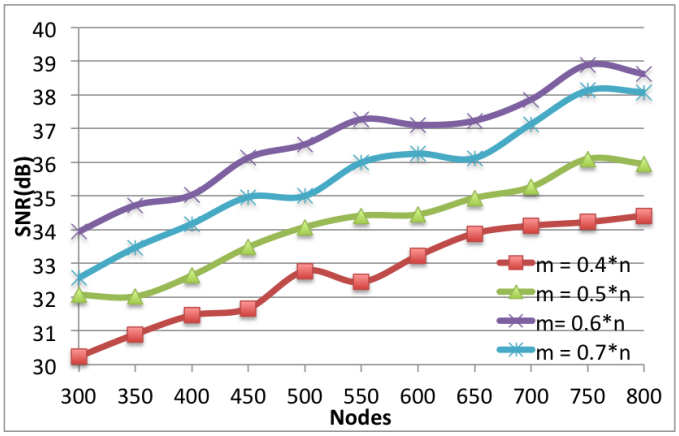


(b) $16 * 16$ *subspaces* for global implementation

Figure 4. An example of the network hierarchy evolution from local interpolation step of NDFT algorithm to global FFT implementation with 500 sensors

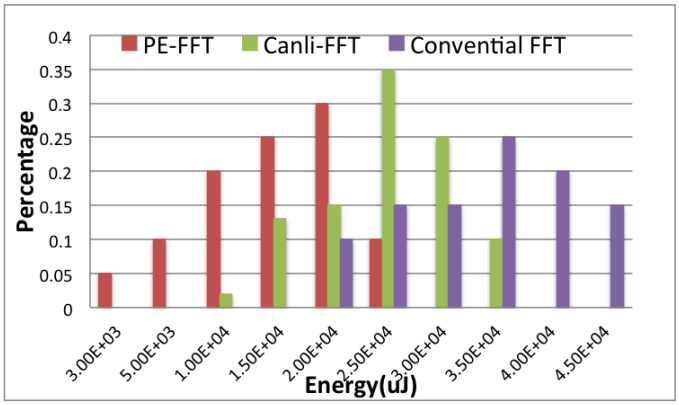


(a) Interpolation step within one cluster

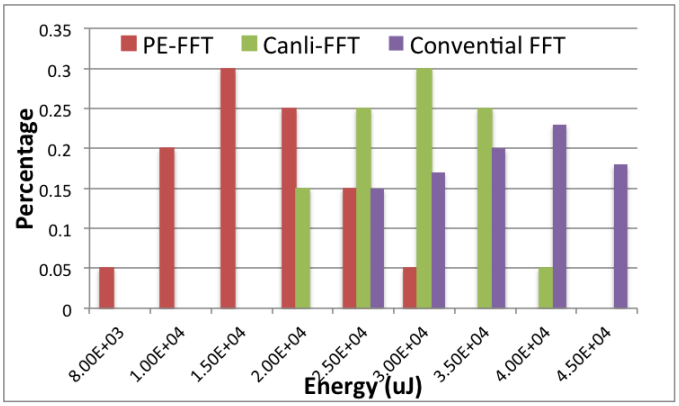


(b) Interpolation step including neighboring clusters

Figure 5. SNR



(a) Network size 400



(b) Network size 600

Figure 6. Histogram of energy consumption for three FFT implementation schemes

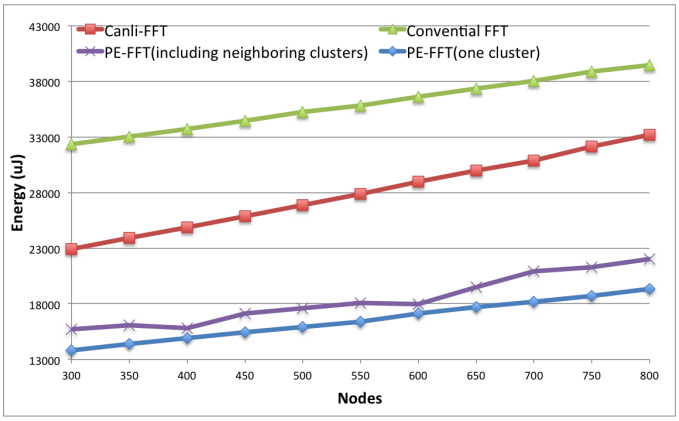


Figure 7. Total energy consumption for all the nodes for three FFT implementation schemes

CHAPTER 3

POWER-EFFICIENT HIERARCHICAL DATA AGGREGATION USING COMPRESSIVE SENSING IN WSNS

3.1 Introduction

Data gathering is the most fundamental task performed within sensor networks. Many environmental monitoring applications, such as monitoring of weather conditions in large parks, the temperature conditions in the ocean, air quality in urban areas, terrain conditions for precision agriculture, and so on, require data measured data sensed at nodes over the entire monitored area to be collected at the fusion center. Most of the existing data collection approaches seek to improve energy efficiency by introducing new protocols in terms of sleep scheduling (19) (20), topology control (21) (22), and mobile data collectors (23) (24). In a conventional data collection scheme, a sensor may seek to transfer one data item to the sink with $N-1$ hops away along a routing path. Each intermediate sensor combines data it receives with its own and forwards it along the route. This data collection scheme without any processing usually entails $O(N^2)$ data transmissions. However, if the monitored area is huge, the cost of direct data forwarding in a large-scale network will be high. Note that the inherent data redundancy – a consequence of the correlated smooth data fields – incurs unnecessary energy consumption in transmission. Data aggregation transmits the data in a compact way by exploiting the spatial correlation in data fields, which essentially extends the functionality of sensor networks as well as improves

the efficiency in terms of computation and communication. Therefore, it complements other approaches by significantly reducing the amount of data to be transported in the routing path.

With the advancement of micro-controller technique, clever utilization of the computational capability of sensors for in-network processing is a new prospect we should explore to overcome the energy limitation in WSNs. Studies in (4) (5) (25) show that power consumed in communication is significantly higher than that in computation. Although many applications for sensor networks are defined and studied, there still remains a huge unexploited region in the application layer of the network stack (2) (3) of sensors.

Compressive sensing (CS) (26) (27) (28), a recently developed signal processing technique, has been under the spotlight in various scientific research communities. The basic principle is that if signals are sparse or compressible in some domain, CS promises to deliver a full recovery of signals with high probability from far fewer samples than the original data. Several research efforts have been pursued to incorporate CS into data collection schemes in WSNs (29) (30) and into spatio-temporal data collection (31) (32). However, the assumption common to all these papers is that each sensor takes a sequence of samples, to which CS is applied to remove temporal redundancy before data transmission starts, which more precisely should be categorized as multi-signal gathering schemes. Luo and coworkers (33) firstly reported the use of Compressive Sensing (CS) in data gathering scheme aiming to remove data redundancy existing in the routing path. We denote their scheme as Plain CS (PCS) aggregation. PCS made a reasonable assumption that the sparsity of the data field is a constant and each data collection cycle gathers one unit of data from all the sensors. First of all, the tasks of sensing and data

propagation in most applications are executed periodically (34). It is easy and feasible to derive the key features from the previous dataset and store them if the size of the key feature set is not big and data field changes slowly within a short time period. In this scenario, the sparsity K of the signal for the entire data field, which denotes the nonzero entries of data or of its transformed version, can be claimed as a *a priori* information. Secondly, multi-signal gathering task can be realized by integrating multiple data collections with different time stamps. PCS requires each sensor to provide to the sink at least $M = K \log N$ measurements so as to fully recover a total of N data samples for the whole field. It reduces the required transmission complexity from $O(N^2)$ to $O(NM)$ in a N-sensor network. However, in the initial phase of PCS, leaf nodes unnecessarily transmit M measurements, which is in excess of their single readings and therefore introduce redundancy in aggregated data. Recognizing this limitation, a hybrid CS (HCS) aggregation scheme was proposed in (35)(36), which is an amalgam of non-CS aggregation and PCS. It optimizes data aggregation cost by setting a global threshold M and applying CS aggregation only at those nodes where the number of accumulated data samples equals or exceeds M; otherwise, it follows the conventional data gathering scheme. Hence transmission waste was eliminated inherent in the initial data transmission. Note that in HCS, only a small fraction of sensors actually can utilize CS because of the inefficient network configuration, and the required amount of data that need to be transmitted for these nodes is still large.

Observing these limitations, we propose an energy-efficient data aggregation technique based on a multi-level hierarchical clustering architecture and hybrid compressive sensing. The central

idea is to configure sensor nodes so that instead of one sink node being targeted by all sensors, several nodes, arranged in a way to yield a hierarchy of multi-level clusters, are designated for the intermediate data collection. The amount of data that needs to be transmitted by each sensor is determined by the local cluster size at different levels rather than the entire network. The main reason for choosing the multi-level hierarchy for communication routing path is two-fold: Firstly, the work in (37) demonstrated that for the common data transformed representation, the power efficiency of data aggregation implemented in the hierarchy is generally better than other data communication protocols. Secondly, multi-level hierarchy applies CS in a way that it enables the amount of CS random measurements to adapt to the size of the clusters at different levels. We refer to our method as Hierarchical Data Aggregation using Compressive Sensing (HDACS).

The main contributions of this work are summarized as follows:

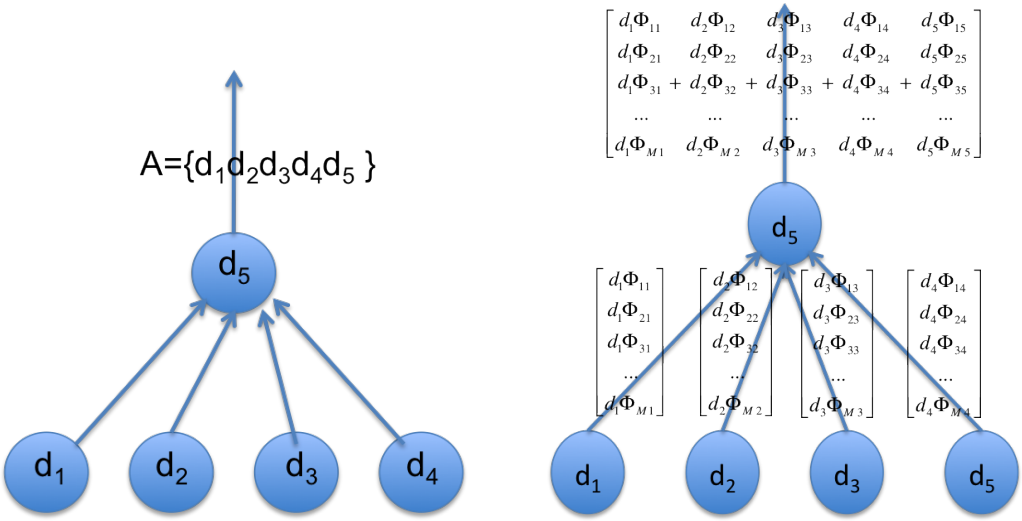
- We formulate a novel data aggregation scheme by incorporating Compressive Sensing (CS) in a multi-level hierarchy for large-scale sensor networks.
- We prove HDACS essentially reduces data volume in transmission compared with other CS-based data aggregation schemes with an upper bound of $O(K \log N)$ and it also achieves the highest data compression ratio.
- We construct a new energy model by delving into the cost in processor and radio, especially for computation cost incurred in relatively complex algorithms, such as CS data recovery. We establish that communication is still a dominant concern in HDACS with the supporting specs from real sensor datasheets.

- We choose a suitable signal model for data field and design a customized DCT-based CS recovery algorithm to adapt to the chosen domain by factoring in computation complexity, speed and accuracy.
- We use the surface temperature dataset from Nation Data Buoy Center as well as synthetic dataset to evaluate the performance of different CS-based data aggregation schemes on SIDnet-SWANS simulation platform. The superiority of the proposed HDACS in terms of data reconstruction quality and power efficiency is effectively validated.

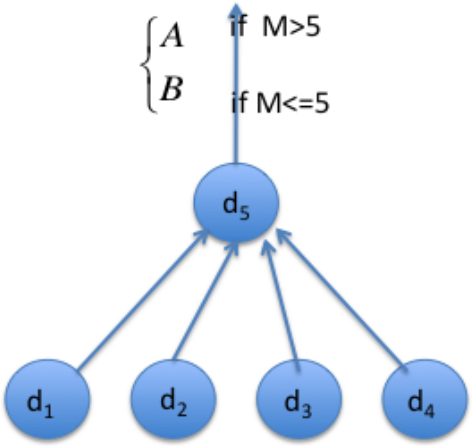
3.2 Related Work

Power efficiency is a key objective for data aggregation scheme in wireless sensor networks. Plain CS (PCS) (33) is a novel CS data aggregation scheme to reduce the amount of data need to be transmitted by integrating CS as a compressing technique and thereby to improve communication power efficiency. CS data aggregation requires each sensor to transmit $M = K \log N$ units of data for a N-sensor network, where K is the signal sparsity of the data field.Transmission of equal amount of data for each sensor regardless of its position and its role in the aggregation routing path introduces redundancy and entails unnecessary energy consumption. In order to solve this problem, a hybrid CS (HCS) aggregation scheme (35) (36) was proposed where sensors apply CS only when the number of data items collected exceeds M; otherwise, sensors combine the received data with their own data and send them directly, which is the same as conventional data aggregation scheme.

Figure 8 shows a simple example of three different data aggregation mechanisms in which nodes $i = 1, 2, 3, 4$ send data d_i to node 5. In a conventional or non-CS (NCS) data aggregation,



(a) NCS (b) PCS



(c) HCS

Figure 8. Example of Different Data Aggregation Mechanisms

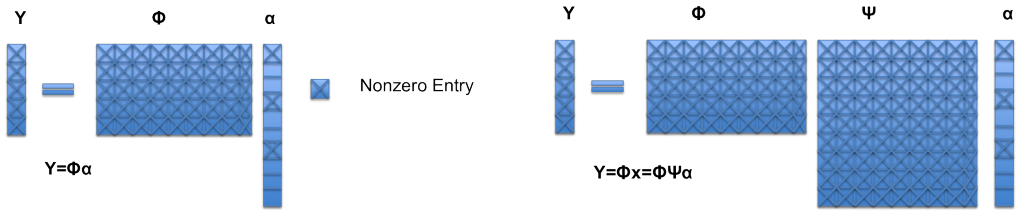
shown in Figure 8(a), node 1 to 4 send their data to node 5 and node 5 combines data it receives with its own data d_5, forms a new data set $A = \{d_1, d_2, d_3, d_4, d_5\}$ and sends A to the next node. In PCS, Figure 8(b) shows each node $i = 1, 2, 3, 4$ takes M randoms CS measurements by multiplying its data with corresponding i^{th} column of sensing matrix Φ to obtain encoded data vector $D_i = [d_i * \Phi_{1i}\ d_i * \Phi_{2i}\ \ldots\ d_i * \Phi_{Mi}]^T$ and then sends them to node 5. Node 5 adds all data received to its own encoded data vector and obtain a new one $B = \sum_{i=1}^{5} D_i$ for transmitting to the next node. For HCS, Figure 8(c) depicts that each node $i = 1, 2, 3, 4$ sends its data d_i to node 5. But whether or not data being encoded for transmission depends on the global threshold M. If $M > 5$, node 5 transmits data set A; otherwise, it transmits B. Compared with NCS and PCS, HCS optimizes the amount of data for transmission by utilizing CS in a selected manner.

3.3 Preliminaries

3.3.1 Compressive Sensing

If a data set $\boldsymbol{\alpha}$ or a suitable unitary transformed version $\mathbf{x} = \Psi\boldsymbol{\alpha}$ with length N has K nonzero entries, we call it a K-sparse signal or K-sparse data. Compressive Sensing (CS) theory demonstrates that only $O(M)$ random measurements are enough to represent the transformed data $\mathbf{x}$ (38) (28) (39) (40), where $K < M << N$, if the random measurements taken from the sensing matrix Φ with size $M \times N$ follows Uniform Uncertainty Principle (UUP) (41) (42):

$$0.8\frac{M}{N}||\mathbf{x}||_2^2 \leq ||\Phi\mathbf{x}||_2^2 \leq 1.2\frac{M}{N}||\mathbf{x}||_2^2.$$



(a) Sparse Data $\boldsymbol{\alpha}$ (b) Sparse Signal $\mathbf{x}$

Figure 9. CS Random Projection

If it is K-sparse data, then by replacing $\mathbf{x}$ with $\boldsymbol{\alpha}$, the bounds above dictated by UUP still apply. Figure 9 shows these two types of CS random measurements processes. This property is very appealing as it reduces data representation from original length N to M, which could be significant when N is large.

The condition of for accurately recovering a signal from CS random measurements is $M >= C\mu^2(\Phi, \Psi)K \log N$, where C is a small number and $\mu^2(\Phi, \Psi)$ is a defined function to measure the mutual coherence between the transformation matrix Ψ and the sensing matrix Φ. Under a certain situation, such as sampling ultra-wideband but spectrally sparse signal, $\mu^2(\Phi, \Psi)$ is equal or close to one, then on the order of $K \log N$ samples suffice (26). Since our work focuses on the relationship between the amount of data transmitted and the scalability of the network, the assumption of $C = 1$ will not alter the comparison results. We hence assert that $M = K \log N$ is a necessary number for delivering data without losing its fidelity in the following paragraph.

Figure 9(a) shows one way of taking CS random measurements, where the data field is itself assumed to be sparse. This applies for various event detection fields, such as fire, traffic, and

TABLE II

GLOBAL PARAMETERS DEFINITION

Global parameters	
N	Total number of nodes in the network
n	The degree of logical tree
s	Area associated with level one cluster
S	The whole area monitored by sensor networks
T	Number of levels in data collection hierarchical tree

smoke, etc. For some other types of data fields which are smooth and slowly varying, such as temperature, sound, pressure fields, Figure 9(b) shows another way to solve the problem. It projects data into a suitable unitary transformation space to firstly obtain sparse data representation and then applies the projected data onto a sensing matrix. Therefore, CS-based data aggregation is applicable in both cases. We focus our study on the second scenario, which is a representative of various environment monitoring applications in sensor networks.

3.4 Proposed Data Aggregation Architecture

In this section, the data aggregation architecture based on compressive sensing is presented. We mainly focus on examining and solving two key problems: 1) How to configure sensors to construct a hierarchical network architecture? 2) How to implement data aggregation on this architecture so that power efficiency can be enhanced?

3.4.1 Model and Aggregation Process

Assume the hierarchy is constructed based on the geographical location and area. At level one, all the clusters are defined by identical regions each with area s. With the increase in

TABLE III

CLUSTER-SPECIFIED && PARAMETERS DEFINITION

Local parameters in cluster l at level i	
$s_i^{(l)}$	Cluster area
$c_i^{(l)}$	Cluster head node
$v_i^{(l)}$	Collection of children cluster heads
$N_i^{(l)}$	Cluster size including all the nodes in this cluster
$M_i^{(l)}$	Amount of data that need to be transmitted after performing CS from cluster head
$d_i^{j(l)}$	Distances between cluster head $c_i^{(l)}$ and its children cluster head $j(l)$, where $j(l) \in v_i^{(l)}$
$\gamma_i^{(l)}$	Data compression ratio at cluster head $c_i^{(l)}$
$E_i^{(l)}$	Total transmission cost from all its children nodes to $c_i^{(l)}$

TABLE IV

LEVEL-SPECIFIED PARAMETERS DEFINITION

Global parameters at level i	
C_i	Collection of cluster heads, where $C_i = \{c_i^{(1)}, c_i^{(2)}, \cdots, c_i^{(\lvert C_i \rvert)}\}$
$\lvert C_i \rvert$	Number of cluster heads
M_i	Total amount of data units for transmission, where $M_i = \sum_{l=1}^{\lvert C_i \rvert} M_i^{(l)}$
E_i	Total energy cost for transmission, where $E_i = \sum_{l=1}^{\lvert C_i \rvert} E_i^{(l)}$

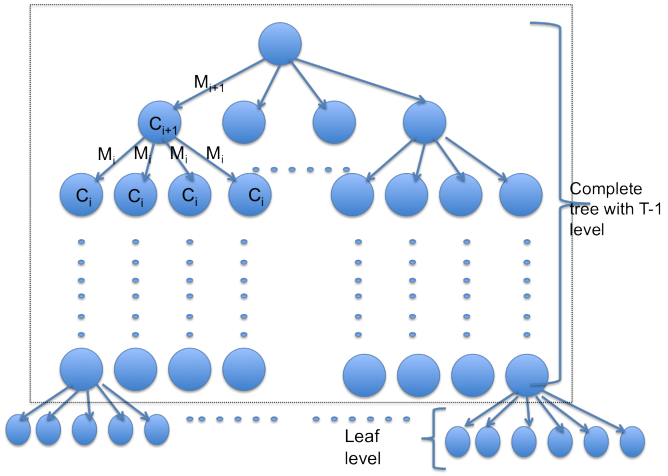


Figure 10. CS Data Aggregation Architecture

level i, where $i > 2$, cluster l is configured by merging the areas of n clusters at level $i-1$, i.e. it satisfies: $s_i^{(l)} = ns_{i-1}^{(l)} = n^{i-1}s$. This construction process guarantees the same area for all the clusters at the same level. Follow this procedure, we get the relation between the entire monitored area S and the initial cluster area s: that is $S = n^{T-1}s$. Let $|C_i|$ represent the number of clusters at level i and we get the relation between the number of clusters and the whole area: $s_i^{(l)}|C_i| = n^{i-1}s|C_i| = S$. so, the number of clusters at level i is $|C_i| = \frac{S}{s}n^{1-i} = n^{T-1}n^{1-i} = n^{T-i}$. For convenience, the definitions of the parameters in our formulation are listed in Table II, Table III, and Table IV.

Figure 10 gives an instance of a hierarchical tree to show the logical relationship among clusters at multiple levels in the hierarchy. The hierarchy consists of n nodes at level i for $i \geq 2$,

or in another words, the degree of the hierarchical tree is n. We also assume the number of leaf nodes $N_1^{(l)} \geq n$ for any cluster l at level one.

The distribution of a N-node network, where $N = N' + n^T$, in our formulation is constrained by the following rule.

- There are at least n nodes in each cluster at level one. Accordingly, at least $n * |C_1| = n * n^{T-1} = n^T$ associated nodes will be randomly placed in a constrained manner.
- The remaining $N - n^T$ sensors are allowed to be randomly distributed following uniform distribution within the entire region.

The main advantage of this network deployment is that it is practically realizable and it also addresses issues when the total number of nodes N does not satisfy the condition $N = n^T$. Besides, we get the number of cluster heads in the hierarchy at most n^{T-1} (If a node plays cluster head role at more than two levels, the number of cluster heads should be less than n^{T-1}). And the number of leaf nodes is $N - n^{T-1} \geq n^T - n^{T-1} = (n-1)n^{T-1}$. If $n > 2$, as a result, $N - n^{T-1} > n^{T-1}$. It implies that the number of leaf nodes is much more than the number of cluster heads. The rules also ensures that the number of node in one cluster at level i satisfies $N_i^{(l)} \in \{n^i, n^i + 1, \ldots, n^{i+1} - 1\}$. In the following sections, we will use these initial settings and conclusions to analyze different parameters of interest.

Algorithms 2, 3, and 4 are the pseudo codes for the behaviors of nodes playing different roles in the hierarchy and they shows the explicit steps of HDACS. Algorithm 2 shows that in phase one, a leaf node sends one unit of data it measures from the environment to its cluster head

$c_1^{(l)}$, which follows the same step as HCS (35) (36) and then its task in this data aggregation is done. Algorithm 3 shows cluster head $c_1^{(l)}$ collects all the data from $N_1^{(l)} - 1$ children nodes, projects them to a suitable space to get a sparse signal representation, and then derives CS random measurements from the sparse signal, which compresses the number of data from $N_1^{(l)}$ into $M_1 = K * \log N_1^{(l)}$. Algorithm 4 describes the general behaviors of cluster heads at level higher than one. When cluster head $c_i^{(l)}$ at the level i higher than 2 receives $M_{i-1}^{(j(l))}$ CS random measurements from its children cluster $j(l)$, where $j(l) \in v_i^{(l)}$ and $v_i^{(l)}$ is a defined collection consisting of all its children nodes, it executes the CS recovery algorithm to first estimate the transformed signal and then retrieve the original data through inverse transformation. After it recovers all the data from the lower level, a cluster head takes $M_i^{(l)} = K \log N_i^{(l)}$ CS random measurements based on $N_i^{(l)}$ units of data. The compressed data will thereby be sent to its parent cluster head at level $i + 1$. Following this procedure, the data flows from the bottom of the hierarchical tree, gets assembled with other data from the cluster heads at different levels all the way up util it arrives at the cluster head in the top level. And data aggregation is finally accomplished.

In the following section, we will provide the analysis of key parameters of interest for the proposed HDACS as well as other state-of-the-art CS-based data aggregation schemes(33) (35) (36).

Algorithm 2 Pseudo Code For Leaf Nodes At Level 1

Initial: level $i = 1$, $sensedData$
Send $sensedData$ to $myClusterHead_1$;

Algorithm 3 Pseudo Code For Cluster Heads At Level 1

Initial: level $i = 1$, $sensedData$
Save $sensedData$;
Receive data from children nodes;
if the data received from $N_1^{(l)}$ nodes **then**
    Get sparse signal representation from the entire data;
    Take CS random measurements from the sparse signal;
    $i++$;
    if $myNode$ is my cluster head at level 2 **then**
        Save CS random measurements;
    else
        Send CS random measurements to cluster head at level 2;
    end if
end if

Algorithm 4 Pseudo Code For Cluster Heads At Level i For $i > 1$

```
Receive data from all children nodes, perform CS recovery algorithm and inverse transformation to obtain original data;
if the data received from n nodes then
    i++;
    if i == (T+1) then
        Store all the data for the entire data field;
        Break;
    else
        Get sparse signal representation for the entire data;
        Take CS random measurements from the sparse signal;
        if myNode is my cluster head at level i then
            Save CS random measurements;
        else
            Send CS random measurements to my cluster head at level i;
        end if
    end if
end if
```

3.5 Parameters Analysis

3.5.1 The Amount of Data That Needs To Be Transmitted At One Level

In the initial network configuration, we assume the number of nodes $N_i^{(l)}$ in the cluster l at the level i should satisfy the condition that $N_i^{(l)} \in \{n^i, n^i + 1, \ldots, n^{i+1} - 1\}$. Accordingly, CS random measurements or the amount of data that needs to be transmitted for a cluster head in HDACS is: $M_i^{(l)} = K \log N_i{}^{(l)} \in [iK \log n, (i+1)K \log n]$.

The required amount of data that needs to be transmitted for each node in PCS (33) is a constant: $M_{i,PCS}^{(l)} = K \log N$. When PCS is implemented in a multi-level hierarchy, this amount $M_{i,PCS}^{(l)}$ will still be the same for all the cluster heads and $M_{i,PCS}^{(l)} \in [TK \log n, (T+1)K \log n]$.

For HCS (35) (36), the required amount of data that needs to be transmitted for each node for a cluster head at level i in cluster l is:

$$M_{i,HCS}^{(l)} = \begin{cases} N_i^{(l)} & \text{if } N_i^{(l)} < K \log N, \\ K \log N & \text{otherwise.} \end{cases}$$

Proposition 4. *The amount of data that needs to be transmitted at any given level i for any cluster l for PCS and HCS in a multi-level data aggregation hierarchy is lower bounded by the amount of data that needs to be transmitted in HDACS.*

Proof. The upper bound for the amount of data that needs to be transmitted for cluster heads at any level in HDACS is $O(M_{i,HDACS}^{(l)}) = M_{i,PCS}^{(l)} = K \log N$. It therefore follows that $M_{i,HDACS}^{(l)} \leq M_{i,PCS}^{(l)}$.

Besides, at level $i > 1$

$$\begin{cases} M_{i,HDACS}^{(l)} = K \log N_i < M_{i,HCS}^{(l)} = N_i^{(l)} & \text{if } N_i^{(l)} < K \log N, \\ M_{i,HDACS}^{(l)} = K \log N_i < M_{i,HCS}^{(l)} = K \log N & \text{otherwise.} \end{cases}$$

and at level $i = 1$, $M_{i,HDACS}^{(l)} = M_{i,HCS}^{(l)} = N_i^{(l)}$. As a result, the inequality $M_{i,HDACS}^{(l)} \leq M_{i,HCS}^{(l)}$ is true at any level.

In summary, HDACS requires less amount of data that need to be transmitted than PCS and HCS at any level in a multi-level data aggregation hierarchy. □

3.5.2 The Total Amount of Data Transmitted

For convenience, we denote $[f(i)]_a^b = \sum_{i=a}^{b} f(i)$ in the following analysis.

Proposition 5. *In a multi-level hierarchy with T levels, the total amount of data transmitted in PCS is: $M_{total,PCS} = NK \log N$, and the space complexities of the total amount of data transmitted in HDACS and HCS are: $M_{total,HDACS} = \Theta(N)$ and $M_{total,HCS} = \Theta(N \log N)$ respectively. Consequently, HDACS requires the lowest amount of data that need to be transmitted compared with the other two CS-based data aggregation schemes.*

Proof. 1. For HDACS, the total amount of data that needs to be transmitted is:

$$\begin{aligned} M_{total,HDACS} &= \sum_{i=1}^{T} M_i = \sum_{i=1}^{T} \sum_{l=1}^{|C_i|} M_i^{(l)} \\ &= \sum_{l=1}^{|C_1|} (N_1^{(l)} - 1) + \sum_{i=2}^{T} \sum_{l=1}^{|C_i|} (n-1) M_{i-1}^{(l)}. \end{aligned}$$

As we assume $N_i^{(l)} \in \{n^i, n^i + 1, \ldots, n^{i+1} - 1\}$, the lower bound of $M_{total,HDACS}$ is:

$$\Omega(M_{total,HDACS}) = N - n^{T-1} + K(n-1)n^{T-1} \log n [in^{-i}]_1^{T-1}$$

and its upper bound is

$$O(M_{total,HDACS}) = N - n^{T-1} + K(n-1)n^{T-1} \log n [(i+1)n^{-i}]_1^{T-1},$$

where $[in^{-i}]_1^{T-1} = \frac{1/n(1-1/n^{T-1})}{(1-1/n)^2} - \frac{T-1}{n^T(1-1/n)} < \frac{1/n}{(1-1/n)^2}$ and $[(i+1)n^{-i}]_1^{T-1} = [in^{-i}]_1^{T-1} + [n^{-i}]_1^{T-1} = \frac{1/n(1-1/n^{T-1})}{(1-1/n)^2} - \frac{T-1}{n^T(1-1/n)} + \frac{1/n*(1-1/n^{T-1})}{1-1/n} < \frac{1/n}{(1-1/n)^2} + \frac{1}{n-1}$ are both constant scalars determined by the initial logical cluster size n. In the initial setting, we get $(n-1)n^{T-1} \approx N$. **Therefore, the space complexity of $M_{total,HDACS}$ is $\Theta(N)$.**

2. For PCS, the total amount of data that needs to be transmitted is: $\boldsymbol{M_{total,PCS} = NK \log N}$.

3. For HCS, the decision of performing CS in cluster heads depends on the value of cluster size $\boldsymbol{N_i^{(l)}}$ and the global threshold $\boldsymbol{K \log N}$.

   Suppose at level $\boldsymbol{i = t}$, the condition $\boldsymbol{N_t^{(l)} \leq K \log N \leq N_{t+1}^{(l')}}$ is satisfied for any particular cluster $\boldsymbol{l}$, which implies that the application of CS will be feasible afterwards. We can easily derive the the following inequality:

$$\boldsymbol{\frac{\log(K \log N)}{\log n} - 2 \leq t < \frac{\log(K \log N)}{\log n}.}$$

   Note that the above inequality shows $\boldsymbol{t}$ is roughly proportional to $\boldsymbol{\log(\log N)}$. It implies $\boldsymbol{t}$ grows very slowly with the increase of network size $\boldsymbol{N}$. Therefore, we can consider it as a negligible number and use a fix constant to represent it.

Consequently, the total amount of data that needs to be transmitted in HCS can be expressed as:

$$\begin{aligned} \boldsymbol{M_{total,HCS}} &= \sum_{i=1}^{t}\sum_{l=1}^{|C_i|}(N_i^{(l)}-1)+\sum_{i=t}^{T}\sum_{l=1}^{|C_i|}(n-1)K\log N \\ &= \sum_{i=1}^{t}(N-n^{T-i})+\sum_{i=t}^{T}n^{T-i}(n-1)K\log N \\ &= Nt-\sum_{i=1}^{t}n^{T-i}+K(n-1)\log N\sum_{i=t}^{T}n^{T-i}. \end{aligned}$$

Therefore, $\boldsymbol{M_{total,HCS} = Nt - n^T[n^{-i}]_1^t + K(n-1)n^T \log N[n^{-i}]_t^T}$. In this formula, $[n^{-i}]_1^t$ and $[n^{-i}]_t^T = [n^{-i}]_1^T - [n^{-i}]_1^{t-1}$ are both scalar constants. Note that the dominant component of $\boldsymbol{M_{total,HCS}}$ is $\boldsymbol{K(n-1)n^T \log N[n^{-i}]_t^T}$, therefore its space complexity is $\boldsymbol{\Theta(N \log N)}$.

In summary, the space complexity of the total amount of data that needs to be transmitted required in HDACS is $\Theta(N)$, whereas it is $\Theta(N \log N)$ in the case of both PCS and HCS. Hence, it proves that HDACS requires the least amount of data that needs to be transmitted compared with other CS-based data aggregation schemes examined in this work. □

3.5.3 Data Compression Ratio

Data compression ratio serves as an important indicator of measuring the reduction in data volume and saving of power spent on data transmission. The data compression ratio $\gamma_i^{(l)}$ for a given cluster l at level i is defined as the ratio of the amount of data available at cluster head $c_i^{(l)}$ to the amount of data that needs to be transmitted.

Proposition 6. *Compared with PCS and HCS, HDACS achieves the highest data compression ratio.*

Proof. 1. For HDACS, the expression of $\gamma_i^{(l)}$ is:

$$\gamma_{i,HDACS}^{(l)} = \begin{cases} \frac{N_1^{(l)}}{M_1^{(l)}} = \frac{N_1^{(l)}}{K \log N_1^{(l)}} & \text{if } i = 1, \\ \frac{\sum_{j(l) \in v_i^{(l)}} M_{i-1}^{j(l)}}{M_i^{(l)}} & \text{if } i \geq 2. \end{cases}$$

From the above formula, we note that the compression ratio $\gamma_{i,HDACS}^{(l)}$ lies in the range: $[\frac{n}{K \log n}, \frac{n^2}{2K \log n}]$ if $i = 1$ and $[n\frac{T-1}{T+1}, n]$ if $i \geq 2$. It implies the amount of data that needs to be transmitted can be significantly compressed to about $\frac{1}{n}$ of the amount of the data contained within cluster heads.

2. For PCS, $\gamma_{i,PCS}^{(l)} = 1$ for any level i, and it shows PCS provides no compression at all.

3. For HCS,

$$\gamma_{i,HCS}^{(l)} = \begin{cases} \frac{N_i^{(l)}}{M_i^{(l)}} = \frac{N_i^{(l)}}{K \log N_i^{(l)}} & \text{when } N_i^{(l)} > K \log N, \\ & \text{satisfied for the first time} \\ 1 & \text{otherwise.} \end{cases}$$

HCS yields compression ratio greater than 1 only when condition $N_i^{(l)} > K \log N$ is met for the first time, otherwise there is no data reduction.

Therefore, we conclude that, for transmission at each level of the hierarchy, HDACS compresses the data more than the other two CS-based data aggregation schemes. □

3.5.4 Energy Consumption Model

In this model, two critical components — processor and radio energy needs — have been considered as the major drains of power consumed in the data aggregation task. Processor energy consumption is incurred in the task of node control, data processing, communication protocol. And radio energy consumption is incurred in the task of receiving and transmitting data package. Other operations such as sensing, I/O, can been considered to be negligible in this scenario.

3.5.4.1 Single Node Energy Consumption Model

For a single node, its energy consumption model is expressed as:

1. Processor:

$$E_P = E_{p,comm} + E_{p,comp} + C_p,$$

where $E_{p,comm}$ indicates the energy consumed within a processor for buffering data received and to be transmitted. On the other hand, $E_{p,comp}$ reflects the energy consumed in data processing. We also use C_p for other power consumptions.

2. Radio:

$$E_R = E_{r,rx} + E_{r,tx} + C_r,$$

where $E_{r,rx}$ and $E_{r,tx}$ indicate the energy consumed in the packet receiving and transmitting respectively. And other costs in radio are summed into a constant C_r.

Therefore, the total power consumption for a single node as a cluster head at one level of the hierarchy is:

$$E = E_{p,comm} + E_{p,comp} + E_{r,rx} + E_{r,tx} + (C_p + C_r).$$

Furthermore, since power in general is calculated as a product of current, voltage and execution time, therefore, we can further express the factors in the above formulate as: $E_{p,comm} = I_p U T_{p,comm}$, $E_{p,comp} = I_p U T_{p,comp}$, $E_{r,rx} = I_{r,rx} U T_{r,rx}$ and $E_{r,tx} = I_{r,tx} U T_{r,tx}$. Since the values of voltage and current for processor and radio can be found in its datasheet for a particular type of a sensor, the relation between execution time and the volume of data becomes our major concern in the following analysis.

3.5.4.2 Data Processing Cost Analysis

In HDACS, each cluster head involves two key data processing tasks: CS random measurements encoding process and CS recovery process. For a general CS encoding process, the multiplication of a $M \times N$ matrix and a $N \times 1$ vector requires $(M \times N + N + M)$ working storage and MN multiplications and $(N-1)M$ additions for computation operations. Therefore, data encoding cost is bounded by $O(NM)$. Besides, CoSaMP is adopted in HDACS to recover CS random measurements, which will be covered in Section 3.6.4. It has been proved that CoSaMP recovers N samples of data from M random measurements using $O(N)$ working storage and $O(N \log N)$ operations for each iteration(43). Therefore, for a cluster head located at cluster l at level i with the cluster size $N_i^{(l)}$ in the hierarchy, we formulate the data processing time as a linear function of the volume of the processed data: $T_{p,comp} = t_p(k_1 N_i^{(l)} M_i^{(l)} + k_2 N_i^{(l)} \log N_i^{(l)})$, where k_1 and k_2 are scalars and t_p is the unit data processing time.

3.5.4.3 Communication Cost Analysis

The communication cost for processor is formulated as a linear function of the amount of data it receives and it sends. To be more specific, data here includes the readings measured from the sensors and the necessary information, such as the indexes of the randomly selected rows of the sensing matrix. We hence formulate it as: $T_{p,comm} = t_p(\sum_{j(l)\in v_i^{(l)}} M_{i-1}^{j(l)} + M_i^{(l)})$, where t_p is the unit data processing time for processor and $M_{i-1}^{(j(l))}$ are the amount of data cluster head $c_i^{(l)}$ receives from its child node $j(l)$ and $M_i^{(l)}$ is the volume of data it sends out. On the other hand, as the packet transmission latency for radio should also be proportional to the amount of data in transmission, we formulate it as: $T_r = t_{r,rx} \sum_{j(l)\in v_i^{(l)}} M_{i-1}^{j(l)} + t_{r,tx} M_i^{(l)}$, where $t_{r,rx}$ is the radio receiving time for unit data and $t_{r,tx}$ is the radio transmitting time for unit data.

Let $W_p = I_p U t_p$, $W_{r,rx} = I_{r,rx} U t_{r,rx}$, $W_{r,tx} = I_{r,tx} U t_{r,tx}$, $C = C_p + C_r$, and these parameters are deemed as constants in the following analysis. When the single node energy model is incorporated into the hierarchy, the energy consumption for a cluster head at level i within cluster l is:

$$E_i^{(l)} = W_p(k_1 N_i^{(l)} M_i^{(l)} + k_2 N_i^{(l)} \log N_i^{(l)}) + (W_{r,rx} + W_p) \sum_{j(l)\in v_i^{(l)}} M_{i-1}^{j(l)} + (W_{r,tx} + W_p) M_i^{(l)} + C.$$

Note that a large-scale network will inevitably involve a long haul data transmission, i.e. cost in a number of relay nodes along the routing path should be taken into account and it involves communication cost in processor as well as radio. As the communication cost for relays

is positively correlated with distance, it is reasonable to model it as a linear function of d_i^{α}, where d_i is the distance between the sender and receiver and α is a power loss exponent (44), where we use $\alpha = 2$ in our case. The total energy consumption for a cluster head $c_i^{(l)}$ at level i in the hierarchy is adjusted as:

$$E_i^{(l)} = c_r(k_1 N_i^{(l)} M_i^{(l)} + k_2 N_i^{(l)} \log N_i^{(l)}) + c \sum_{j(l) \in v_i^{(l)}} (d_i^{j(l)})^2 M_{i-1}^{j(l)} + c_s,$$

where we define $c_r = W_p$, $c = W_{r,rx} + W_{r,tx} + W_p$ and $c_s = C$.

Define the distance $d_i^{j(l)}$ between the cluster head $c_i^{(l)}$ and its children cluster head $j(l)$ for $j(l) \in v_i^{(l)}$ as: $d_i^{j(l)} = ((x_j - x_i)^2 + (y_j - y_i)^2)^{1/2}$, where (x_i, y_i) and (x_j, y_j) are the coordinates of cluster head $c_i^{(l)}$ and children node $j(l)$ respectively. For brevity, we denote the locations of cluster head (x_{c_i}, y_{c_i}) and children nodes $(x_{j(l)}, y_{j(l)})$ with (x_i, y_i) and (x_j, y_j) respectively. The sum of squared distances between a cluster head and its children cluster heads needs to be estimated to compute energy consumption. We will model the sensor locations as randomly distributed and use the expected value D_i of the sum of squared distances as an estimate for use in computing energy costs. Let $D_i = \sum E[(X_j - X_i)^2 + (Y_j - Y_i)^2]$, where the coordinates X_j, X_i, Y_i, Y_j are assumed to be uniformly distributed. Since all children nodes are identically distributed, $D_i = (n-1)E[(X_j - X_i)^2 + (Y_j - Y_i)^2]$ for any j.

Proposition 7. *The sum of squared distance D_i between the cluster head $c_i^{(l)}$ and all its children cluster heads $j(l) \in v_i^{(l)}$ is in the range of $[(n-1)sn^{i-1}/6,\ 2(n-1)sn^{i-1}/3]$.*

Proof. For randomly distributed sensors, D_i can be expressed as:

$$\begin{aligned} D_i &= (n-1)E[(X_j - X_i)^2 + (Y_j - Y_i)^2] \\ &= (n-1)\int\int\int\int ((x_j - x_i)^2 + (y_j - y_i)^2) f(x_i, y_i) f(x_j, y_j) \mathrm{d}x_i \mathrm{d}y_i \mathrm{d}x_j \mathrm{d}y_j \end{aligned}$$

.

Let $g(x_i, y_i) = \int\int ((x_j - x_i)^2 + (y_j - y_i)^2) f(x_j, y_j) \mathrm{d}x_j \mathrm{d}y_j$ denote the expected value of the sum of the squared distance between cluster head and its children nodes when the location of cluster head is given. Partial derivative in terms of x_i is $\frac{\partial g(x_i, y_i)}{\partial x_i} = 2\int\int (x_i - x_j) f(x_j, y_j) \mathrm{d}x_j \mathrm{d}y_j$. We conclude: if $x_i > E[x_j]$, then $\frac{\partial g(x_i, y_i)}{\partial x_i} > 0$; if $x_i < E[x_j]$, then $\frac{\partial g(x_i, y_i)}{\partial x_i} < 0$; if $x_i = E[x_j]$, then $\frac{\partial g(x_i, y_i)}{\partial x_i} = 0$. The same result applies for y_i. Clearly, it suffices to say the function $g(x_i, y_i)$ has paraboloid shape and it obtains its minimum value when $x_i = E[x_j]$ and $y_i = E[y_j]$, which is exactly the centroid of a cluster. This function achieves its maximum value when the coordinates of the cluster head are at any of the four corners, where the average distance between the cluster head and its children nodes is the greatest.

Let b_i denote half the side length of a square region at level i , i.e. $b_i = 1/2 s_i^{1/2} = 1/2(sn^{i-1})^{1/2}$, where s_i is the area of cluster i and s is the unit area defined at level one. The density function is $f(x_i, y_i) = f(x_j, y_j) = \frac{1}{s_i} = \frac{1}{sn^{i-1}}$.

1. When the cluster head is the centroid, we get the lower bound:

$$\begin{aligned}
&\sum_{j(l)\in v_i^{(l)}} (x_j - x_i)^2 + (y_j - y_i)^2 \\
= \;& 4\int_0^{b_i}\int_0^{b_i} (x_j^2 + y_j^2) f(x_j, y_j) \mathrm{d}x_j \mathrm{d}y_j \\
= \;& 8\frac{n-1}{sn^{i-1}} \int_0^{\pi/4}\int_0^{b_i \sec\theta} \mu^2 \mu \mathrm{d}\mu \mathrm{d}\theta \\
= \;& 8\frac{n-1}{sn^{i-1}} \int_0^{\pi/4} \frac{1}{4} b_i^4 \sec^4\theta \mathrm{d}\theta \\
= \;& 2\frac{n-1}{sn^{i-1}} (1/2(sn^{i-1})^{1/2})^4 (\frac{1}{3}\tan\theta^2 + \frac{2}{3}\tan\theta)|_0^{\pi/4} \\
= \;& \frac{(n-1)sn^{i-1}}{6}.
\end{aligned}$$

2. When the cluster head is located at one of the four corners, we get the upper bound:

$$\begin{aligned}
&\sum_{j(l)\in v_i^{(l)}} (x_j - x_i)^2 + (y_j - y_i)^2 \\
= \;& \int_0^{2b_i}\int_0^{2b_i} (x_j^2 + y_j^2) f(x_j, y_j) \mathrm{d}x_j \mathrm{d}y_j \\
= \;& 2\frac{n-1}{sn^{i-1}} \int_0^{\pi/4}\int_0^{2b_i \sec\theta} \mu^2 \mu \mathrm{d}\mu \mathrm{d}\theta \\
= \;& \frac{2(n-1)sn^{i-1}}{3}.
\end{aligned}$$

In summary, the sum of the squared distance $\sum_{j(l)\in v_i^{(l)}} (d_i^{j(l)})^2$ between cluster head $c_i^{(l)}$ and all its children cluster heads is in the range of $[(n-1)sn^{i-1}/6,\ 2(n-1)sn^{i-1}/3]$. □

Based on Proposition 7, we let $cD_i = A(n-1)n^{i-1}$, where A is a communication cost related constant and $A \in [cs/6, 2cs/3]$.

3.5.4.4 Total Energy in the Hierarchy

The total amount of energy consumed for data aggregation from the bottom to the top of the hierarchy in HDACS is therefore:

$$\begin{aligned} E_{total,HDACS} &= \sum_{i=1}^{T}\sum_{l=1}^{|C_i|} E_i^{(l)} = \sum_{l=1}^{|C_1|} c_s + A(N_1^{(l)} - 1) \\ &+ \sum_{i=2}^{T}\sum_{l=1}^{|C_i|} c_r(k_1 N_i^{(l)} M_i^{(l)} + k_2 N_i^{(l)} \log N_i^{(l)}) + AM_{i-1}^{(l)}(n-1)n^{(i-1)} + c_s \end{aligned}$$

It is worth noting that when the level i is greater than 2, two key components: the communication cost and the computation cost are reflected in the term $\sum_{i=2}^{T}\sum_{l=1}^{|C_i|} AM_{i-1}^{(l)}(n-1)n^{(i-1)}$ and the term $\sum_{i=2}^{T}\sum_{l=1}^{|C_i|} c_r(k_1 N_i^{(l)} M_i^{(l)} + k_2 N_i^{(l)} \log N_i^{(l)})$ respectively. In the following proposition, we prove that under certain circumstance, the amount of energy consumed in HDACS is less than that consumed in PCS and HCS.

Proposition 8. *Define*

$$c_{dif,HDACS,PCS}(N) = \frac{\sum_{l=1}^{|C_1|} A(N_1^{(l)} - 1)(K\log N - 1) + \sum_{i=2}^{T}\sum_{l=1}^{|C_i|} A(n-1)n^{(i-1)}(K\log N - M_{i-1}^{(l)})}{\sum_{i=2}^{T}\sum_{l=1}^{|C_i|} k_1 N_i^{(l)} M_i^{(l)} + k_2 N_i^{(l)} \log N_i^{(l)} - k_1 K \log N}$$

and

$$c_{dif,HDACS,HCS}(N) = \frac{\sum_{i=2}^{t}\sum_{l=1}^{|C_i|} A(n-1)n^{i-1}(N_{i-1}^{(l)} - M_{i-1}^{(l)}) + \sum_{i=t}^{T}\sum_{l=1}^{|C_i|} A(n-1)n^{(i-1)}(K\log N - M_{i-1}^{(l)})}{\sum_{i=2}^{t}\sum_{l=1}^{|C_i|} k_1 N_i^{(l)}(M_i^{(l)} - 1) + k_2 N_i^{(l)} \log N_i^{(l)} + \sum_{i=t}^{T}\sum_{l=1}^{|C_i|} k_1 N_i^{(l)} M_i^{(l)} + k_2 N_i^{(l)} \log N_i^{(l)} - k_1 K \log N}$$

. If $c_r < \min\{c_{dif,HDACS,PCS}\}$ the total energy consumed in HDACS is less than that of PCS; if $c_r < \min\{c_{dif,HDACS,HCS}\}$, the total energy consumed in HDACS is less than that of HCS.

Proof. 1. In PCS, data processing involves encoding a fixed amount of CS random measurements with size $K \log N$, which will be communicated between nodes. Therefore, the total energy consumption of PCS is easily obtained as:

$$\begin{aligned} E_{total,PCS} &= \sum_{l=1}^{|C_1|} c_s + AK \log N(N_1^{(l)} - 1) \\ &+ \sum_{i=2}^{T} \sum_{l=1}^{|C_i|} c_s + AK \log N(n-1)n^{(i-1)} + c_r k_1 K \log N. \end{aligned}$$

In order to compare the energy consumption of HDACS and that of PCS, we compute the difference between $E_{total,HDACS}$ and $E_{total,PCS}$:

$$\begin{aligned} E_{total,PCS} - E_{total,HDACS} &= \sum_{l=1}^{|C_1|} A(N_1^{(l)} - 1)(K \log N - 1) \\ &+ \sum_{i=2}^{T} \sum_{l=1}^{|C_i|} A(n-1)n^{(i-1)}(K \log N - M_{i-1}^{(l)}) \\ &- \sum_{i=2}^{T} \sum_{l=1}^{|C_i|} c_r(k_1 N_i^{(l)} M_i^{(l)} + k_2 N_i^{(l)} \log N_i^{(l)} - k_1 K \log N). \end{aligned}$$

Observe that the term $\sum_{l=1}^{|C_1|} A(N_1^{(l)}-1)(K \log N-1)+\sum_{i=2}^{T} \sum_{l=1}^{|C_i|} A(n-1)n^{(i-1)}(K \log N-M_{i-1}^{(l)})$ actually shows the communication cost comparison between two data aggregation

schemes and $\sum_{i=2}^{T}\sum_{l=1}^{|C_i|} c_r(k_1 N_i^{(l)} M_i^{(l)} + k_2 N_i^{(l)} \log N_i^{(l)} - k_1 K \log N)$ represents the computation cost comparison.When $c_r < \min\{c_{dif,HDACS,PCS}\}$, $E_{total,PCS} - E_{total,HDACS} > 0$ will always be true.

2. In HCS, communication cost for a cluster head at level i in cluster l is proportional to $N_i^{(l)}$ amount of data when data volume is smaller than a global threshold and $K \log N$ thereafter. The data processing cost can be computed in an analogous manner. Therefore, the total energy consumed in HCS is summarized as follows:

$$
\begin{aligned}
E_{total,HCS} &= \sum_{l=1}^{|C_1|} c_s + A(N_1^{(l)} - 1) \\
&+ \sum_{i=2}^{t}\sum_{l=1}^{|C_i|} c_s + AN_{i-1}^{(l)}(n-1)n^{(i-1)} + c_r k_1 N_i^{(l)} \\
&+ \sum_{i=t}^{T}\sum_{l=1}^{|C_i|} c_s + AK\log N(n-1)n^{(i-1)} + c_r k_1 K \log N.
\end{aligned}
$$

And the difference between $E_{total,HDACS}$ and $E_{total,PCS}$ is:

$$
\begin{aligned}
E_{total,HCS} - E_{total,HDACS} &= \sum_{i=2}^{t}\sum_{l=1}^{|C_i|} A(n-1)n^{(i-1)}(N_{i-1}^{(l)} - M_{i-1}^{(l)}) \\
&- c_r(k_1 N_i^{(l)}(M_i^{(l)} - 1) + k_2 N_i^{(l)} \log N_i^{(l)}) \\
&+ \sum_{i=t}^{T}\sum_{l=1}^{|C_i|}(n-1)n^{(i-1)} A(K\log N - M_{i-1}^{(l)}) \\
&- c_r(k_1 N_i^{(l)} M_i^{(l)} + k_2 N_i^{(l)} \log N_i^{(l)} - k_1 K \log N).
\end{aligned}
$$

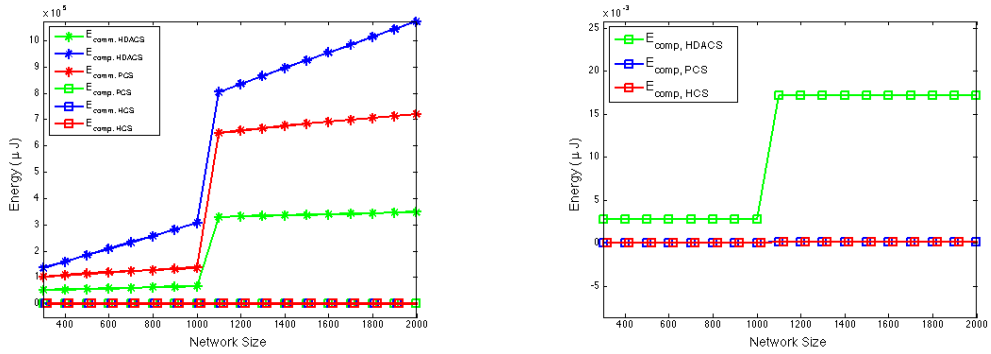
When $c_r < \min\{c_{dif,HDACS,HCS}\}$, $E_{total,HCS} - E_{total,HDACS} > 0$ will always be true.

In summary, if $c_r < \min\{c_{dif,HDACS,PCS}\}$, the total energy consumed in HDACS is less than that of PCS; if $c_r < \min\{c_{dif,HDACS,HCS}\}$, the total energy consumed in HDACS is less than that of HCS. □

Additionally, if we divide the second integral term of denominator and numerator in $c_{dif,HDACS,PCS}$ into two parts at the point $i = t$, we can easily conclude that $c_{dif,HDACS,PCS} \geq c_{dif,HDACS,HCS}$ is true according to the condition that $N_i < K \log N$ when $i < t$. Note that the values of c_r, $c_{dif,HDACS,PCS}$ and $c_{dif,HDACS,HCS}$ are determined by the hardware, we incorporate the hardware specs in the following section to demonstrate that $c_r < \min\{c_{dif,HDACS,HCS}\}$ always holds in the real applications and the data aggregation in HDACS is more power-efficient than that of PCS and HCS.

3.5.4.5 Energy Model Applied in the Hardware

In this section, we choose Mica2 as a typical example. We use the specs from Mica2 (45) with power supply $U = 3(V)$, CPU active current $I_p = 8(mA)$, message receiving current $I_{r,rx} = 7(mA)$, and message transmitting current $I_{r,tx} = 17.4(mA)$. Additionally, Mica2 uses $7.3MHz$ Atmel Microcontroller (46) gives us the single instruction processing time: $t_p = 0.137(\mu s)$. Consistent with IEEE 802.15.4 for low-rate wireless personal area networks, channel bandwidth is set as 38.4 Kbps and thereby $t_{r,rx} = 26(\mu s)$, and $t_{r,tx} = 26(\mu s)$. With the suitable choice of the values of the constant parameters, $\min\{c_{r,HCS}\}$ is computed to be on the order of -2, which is significantly larger than of c_r, which is computed to be on the order of -9. Figure 11(a) plots the results derived from the energy model in terms of the communication and computation



(a) communication and computation cost (b) computation cost

Figure 11. Theoretical results of energy consumptions in terms of communication and computation cost

cost for different CS-based data aggregation schemes. Observe that the communication cost for three schemes are in the order of 5 compared with computation cost in the order of -3. It proves that computation cost is insignificant in these practical applications compared with the communication cost. Note that the number in this graph may not reflect the real value because of the unknown values of the parameters, but our major attention focuses on the trending results. Among the three CS-based data aggregation communication cost, the results demonstrate the superiority of HDACS in energy efficiency even though HDACS requires a bit higher computation cost showed in Figure 11(b). Furthermore, we notice a big jump in communication cost when network scales up to a value, which is between 1000 and 1200. This phenomenon may be attributed to an increase of the network volume to a size that leads to an increment in the total number of levels in the hierarchy.

3.6 Performance Evaluations

In order to perform a comprehensive evaluation, we have used the real-world dataset measuring the sea surface temperature from the Nation Data Buoy Center as well as synthetic data to test our proposed scheme by using the Java-based SIDnet-SWANS(16) simulator and compare the performance with other CS-based data aggregation schemes.

3.6.1 Data Field

We have collected sea surface temperature data from the Nation Data Buoy Center measured by 25 nodes across the Pacific Ocean from 137 E to 95 W, 8N to 8S on April 22, 2014. Figure 12 shows the temperature field after Delaunay triangulation, in which the data is in the range of 27 to 31, which serves as a underlying data field for surveillance. Data value for each monitoring node is obtained by averaging the nearest three data values from the dataset. Considering the limited scale of sensor networks in the real-world applications, we also have tested all the schemes on the synthetic data fields, in which each node observes a constant value corrupted with two types of additive noises: one follows uniformly distribution with zero mean and variance 1/3, and the other is Gaussian noise with zero mean and unit variance.

3.6.2 Signal Sparse Basis

As we target the applications of sensor networks at data fields with smooth and slowly varying property that are densely sampled, such as temperature, humidity, chemical substance, etc, Discrete Cosine Transform (DCT) should be a suitable choice for sparse signal representation. It not only yields fast vanishing moments for signal representation, but also avoids complex coefficients like those in Discrete Fourier Transform (DFT) representations. Complex

computations inevitably increase the algorithm complexity and unnecessarily introduce extra computation cost. Furthermore, many traditional transformations require that the cardinality of data set has to be a power of 2 such as Discrete Wavelet Transform (DWT); otherwise, they need to append more samples i.e. zeros to meet this condition. But DCT does not have such restriction.

Many real-world signals have power-law decline property in coefficients instead of the exact K-sparsity representation (47). In other words, the coefficients decay rapidly as K goes to infinity. To be more precise, such signals are compressible but not K-sparse. Considering this fact, we introduce the signal truncation process so as to obtain K-sparse signals within an allowable error bound to satisfy the prerequisite of compressive sensing. In the simulation, we forced the magnitudes of DCT coefficients smaller than 10% of the first dominant magnitude to zero.

3.6.3 Sensing Matrix

Noiselets(48) known as "noise-like", or in particular, totally un-compressible, is deemed to be incoherent with the most conventional transformations. Its entries are constructed via a multi-scale iteration in a way the same as orthogonal wavelets and wavelet package methods.The self-productive and scalable property make Noiselets as a good candidate as the sensing matrix in the data aggregation hierarchy, which avoids the sensing matrix transmission for assisting CS recovery process. Instead, only the indexes of the randomly elected row of the sensing matrix need to be sent, which may incur an extra $\Theta(M)$ space depending on how they are packed in the message and the size is less than or equal to the scale of data measurements.

3.6.4 CS Recovery Algorithm

In a multi-scale hierarchy, the errors from inaccurate results will be propagated and amplified in the routing path from the bottom to the top of the hierarchical tree, therefore, the accuracy of reconstructed data at each level plays a key role for a robust and effective CS data aggregation scheme. Besides, for an electronic device with limited computation capability, a simple and efficient algorithm is always desirable. All these constraints propel us to find a suitable CS recovery algorithm with high recovery accuracy and low computation cost.

CoSaMP algorithm (47) (43), a convex optimization-based approach, has been proved to have fewer iteration steps than the other Orthogonal Matching Pursuit (OMP) (49) (50) algorithms. Inspired by Uniform Uncertainty Principle (UUP), CoSaMP utilizes the proxy $\mathbf{y} = \Phi * \Phi\mathbf{x}$, which preserves the energy of K largest components of signal $\mathbf{x}$, to approximate the K-sparse signal $\mathbf{x}$.

CoSaMP can identify the location of the largest K nonzero entries of signal $\mathbf{x}$ with a very high probability. Model-based Compressive Sensing (MCS) (47) incorporates signal structural information into CoSaMP and it significantly improves the recovery accuracy. With the aid of the signal structural information, the algorithm shrinks the search space and identifies the location of the largest K nonzero entries in a faster way and essentially reduces the computational complexity. Adopting the same concept as MCS, we design a customized CS recovery algorithm by investigating the characteristics of DCT signal structure to speed up the algorithm to converge to the solution.

Algorithm 5 shows the pseudo code of DCT-based CoSaMP. In each iteration, the algorithm first identifies the maximal $2K$ components, and then uses DCT signal structure information to remove the incorrect identification.The residual in each iteration is preserved as an unsolved part and is utilized to obtain the remaining magnitude in the next iteration. Note that $\Phi^T u$ in Algorithm 5 on line 9 is a product of pseudo inverse of matrix Φ and u. The iteration repeats until it meets either one of termination criteria: the energy of the residue is within a certain bound or the algorithm goes beyond the maximum allowed iterations.

Algorithm 5 DCT-based CoSaMP

1: **Input:** Sensing matrix Φ, CS random measurements u, sparsity K
2: **Output:** A K sparse approximation $\hat{\mathbf{x}}$ of the target signal $\mathbf{x}$
3: $\hat{\mathbf{x}}^0 \leftarrow 0$, $v \leftarrow u$, $k \leftarrow 0$ ▷ Initialization
4: **repeat**
5: $i \leftarrow i + 1$
6: $y \leftarrow \Phi * v$ ▷ Form signal proxy
7: $W \leftarrow supp(y_{2K})$ ▷ Identify large components
8: $T \leftarrow W \bigcup supp(\hat{\mathbf{x}}^{i-1})$ ▷ Merge supports
9: $b|_T \leftarrow \Phi^T u$ ▷ Signal estimation
10: $b|_{T^c} \leftarrow 0$
11: $\hat{\mathbf{x}}^i \leftarrow DCT(b_K)$ ▷ Using DCT structure to prune signal
12: $v \leftarrow u - \Phi\hat{\mathbf{x}}^i$ ▷ Update current samples
13: **until** halting criterion *true*

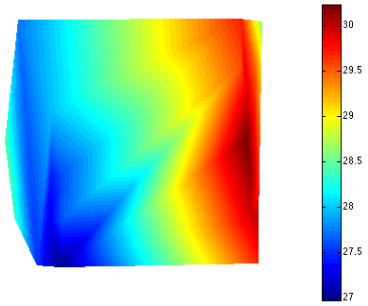


Figure 12. Surface temperature data across the Pacific Ocean from 137 E to 95 W, 8N to 8S on April 22, 2014

3.6.5 Simulation Settings

We have implemented all CS-based data gathering schemes on SIDnet-SWANS (16), (Simulator and Integrated Development Platform for Sensor Networks Applications) to study their performance. JiST/SWANS (Java in Simulation Time) (17) (18),is a Java-based discrete-event simulation engine, where all behaviors of a sensor are simulated and their relevant information can be obtained. Meanwhile, we use SIDnet, a Java-based visual tool, to observe the run-time behaviors of the sensors. A snapshot of user interface of CS-based data aggregation hierarchy on SIDnet-SWANS for 400 sensors network is shown in Figure 13.

In the simulation, the field size is set as $4000 * 4000m^2$, the sensors deployed in the field is in the range of 300 to 800 with 50 nodes as an increments each time, therefore, the average node density increases from $18.75/\text{km}^2$ to $50/\text{km}^2$. We also set the degree of the cluster n as 4. Moreover, the leaf nodes in the first level are assumed to be one hop distance within

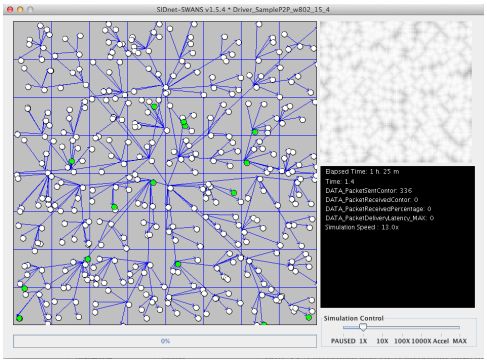


Figure 13. Hierarchical Structure for Network Size 400 on SIDnet-SWANS platform

their cluster. Additionally, the bandwidth is set as 40000bps, which follows IEEE 802.15.4/4a for the low data rate wireless personal data networks and a standard length of one message is 133 bytes (88 bytes for the payload and 45 bytes for control information). Theoretically, the transmission delay is $133 * 8/40000 = 26.6$ms. For simplicity, we neglect the noise interference from the channel and also ignore the possibility of random packet loss. For the setting of the power consumption parameters, we adopt the Mica2 Motes specs again. The default unit transmission cost is set $81\mu J/ms$ per message. We use the same parameter assignments as listed in Table I.

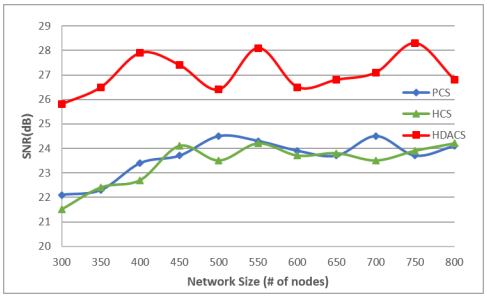


Figure 14. Signal Recovery Results for Real Datasets

3.6.6 Simulation Results

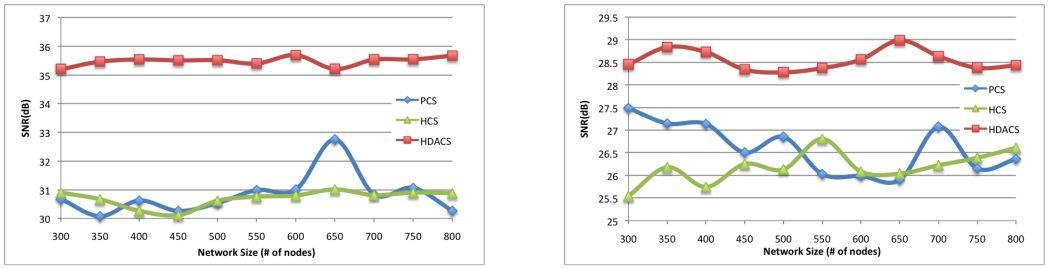
3.6.6.1 Signal Recovery Performance

We define the Signal to Noise Ratio (SNR) as the decimal logarithm of the ratio of the sum of data energy sensed by each sensor over the data recovery error power in the cluster head at the top level. The expression is as follows:

$$SNR_{db} = 10 \log \frac{\sum_{m=0}^{N} ||\boldsymbol{X}_m||_2}{\sum_{m=0}^{N} ||\boldsymbol{X}_m - \hat{\boldsymbol{X}}_m||_2},$$

where $\boldsymbol{X_m}$ is the value sensed by the sensor m and $\hat{\boldsymbol{X}}_{\boldsymbol{m}}$ is the ultimate value collected in the cluster head on the top of the hierarchy.

Figure 14 and Figure 15 show data recovery results for the real datasets and the synthetic datasets versus different CS-based data aggregation schemes, respectively. Both figures show HDACS achieves higher SNR than other schemes. The overall SNR of synthetic dataset (Figure 5) is higher than that of the real datasets (Figure 14). And the overall SNR of data field



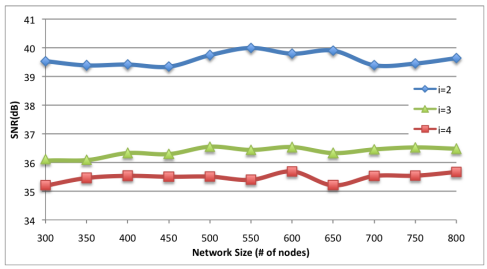
(a) Constant Data Field with Uniformly Distributed Noise

(b) Constant Data Field with Gaussian Noise

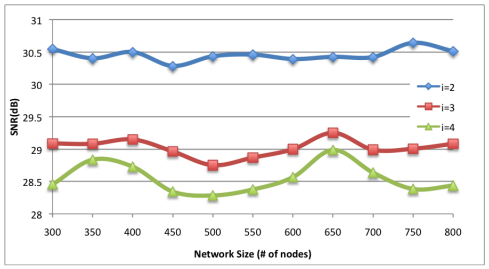
Figure 15. Signal Recovery Results for the Synthetic Datasets

with uniform random noise (Figure 15(a)) is higher than that of the data field with Gaussian noise (Figure 15(b)). We conjecture that this difference is due to data fluctuations in real-world dataset exceeding those in synthetic data. Besides, PCS and HCS tend to have very similar values, which may be the outcome of their similar CS random measurements encoding procedure and the same centralized CS recovery process. Furthermore, signal truncation in PCS and HCS in a cluster at each level is taken based on the whole network size N, which inevitably introduces much higher quantization errors than the decentralized CS recovery algorithm performed in HDACS.

The extra bonus from the hierarchy is that data can be queried locally, i.e. the cluster head at the top level is not the only node where we can query data. Cluster heads at different levels own the entire data from sensors within their area. Therefore, the global data field can also be represented by the union of data obtained from all the cluster heads at any level. Figure 16 shows the data recovery results for two synthetic data fields at different levels, and the graphs

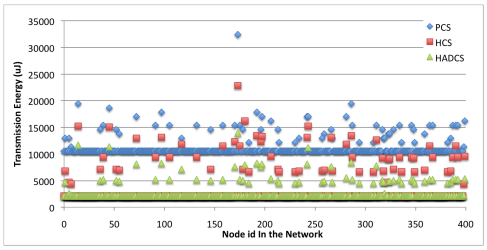


(a) Constant Data Field with Uniformly Distributed Noise

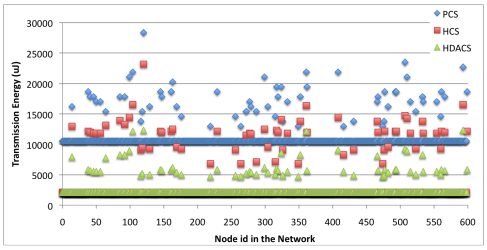


(b) Constant Data Field with Gaussian Noise

Figure 16. Signal Recovery Results from Each Level



(a) Network Size 400



(b) Network Size 600

Figure 17. Transmission Energy Cost Distribution for Different Network Sizes

demonstrate the data recovery performance deteriorates as the level goes up in the hierarchy. (The results of the real datasets shows the same trend.) The reason we proffer is that the errors from signal truncation as well as CS recovery algorithm introduced at each level propagate and are amplified in the hierarchical routing path.

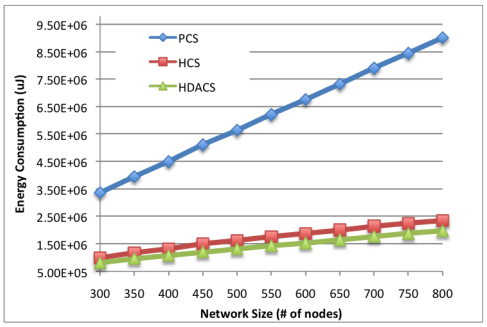


Figure 18. Total Energy Consumption for Different Network Size

3.6.6.2 Energy Consumption

Figure 17 shows the energy consumption for each sensor in a 400-node and 600-node network respectively. Compared to PCS, HDACS saves 34.4% to 80% energy for individual sensors, depending on their roles in the hierarchy. Similarly, for a 600-node network, HDACS saves the energy for nodes approximately 44.5% to 80%. Compared to HCS, HDACS saves at most 55% of energy for nodes in both networks. This result demonstrates the superiority of HDACS in energy efficiency for an individual sensor.

The total energy consumption versus network sizes is shown in Figure 18. Among three CS-based schemes, PCS consumes the most energy. Fixed amount of CS random measurements in transmission for each sensor, regardless of the roles of nodes in the hierarchy severely deteriorates the performance and contributes to the waste of energy. When the network size expands, this problem becomes more pronounced. We also find that HDACS outperforms HCS and energy saving is around 17.88% percentage of HCS. The simulation results shown in Fig-

ure 17 and Figure 18 also demonstrate the effectiveness of the energy model we built in the previous section.

3.6.6.3 Other Related Issues Discussion

Overheads In HDACS, the multi-level clusters are constructed in a way that nodes register themselves to their immediate ancestor nodes from bottom to top following a tree with degree n. This registration overheads come at each level of the hierarchy configuration stage. In the meantime, we assume the hierarchy is built in a static manner and will be utilized for multiple data collections purpose. In this case, the registration overheads for each data collection will be insignificant when the energy expenditure is averaged over all the data collection rounds.

Delay Delay is a common issue in all data collection schemes. In a wireless environment, a nonhierarchical scheme may add $O(N)$ delay due to channel collision issues. In a hierarchical routing scheme, data aggregation starts from the leaf nodes and proceeds toward the sink level by level. Suppose data collection time at each level is a constant D, delay is determined by the product of D and the number of levels $T = \log_n^N$ of the hierarchy. It implies the delay is bounded by $O(\log_n^N)$, which performs generally better than other routing protocols with delay $O(N)$.

Energy Consumption Unfairness The unfairness in energy consumption among nodes is another important issue, which drives a lot of research investigations (51) (52) (53). In order to handle this issue in WSNs, we pursue a simple and efficient localized solution. In our solution the cluster heads with battery charge below a certain level, (e.g., 50 percent-

age) are replaced by the geographically nearest **leaf** nodes within each cluster. Besides, the cluster head node also need to notify its immediate parent node and immediate children nodes about this update. No further communication is required. Each cluster head replacement is performed in such a way that it minimizes the number of nodes involved and so does the related overheads. The proposed solution does not require reorganization of the entire network.

CHAPTER 4

POWER-EFFICIENT NONUNIFORM 2-D FOURIER ANALYSIS USING COMPRESSIVE SENSING IN WSNS

4.1 Introduction

In chapter 2, we (54) proposed a power-efficient Fourier analysis in a 2D random wireless sensor network by adopting the methodology of Nonuniform Discrete Fourier Transform (NDFT) (55) (56)(57)(13). It addressed the problem by designing a distributed hybrid structure consisting of local interpolation step of NDFT algorithm and global FFT computation. A suitable choice of clusters is formed to achieve the initial Fourier coefficients within allowable estimation error bounds. Then separable global 2D FFT is performed along rows and columns to get the final Fourier coefficients. We note that the ratio of the cost of data communication to computation is high, of the order 20:1. Therefore, communication efficiency is a major concern when designing power-efficient algorithms to implement distributed applications in WSNs. In our earlier work (54), we find that energy spent in global communication during FFT dominates the overall energy costs. Traditional separable 2D FFT for computing DFT holds a lot of benefits as it reduces the computation complexity from $O(N^3)$ to $O(N^2 \log N)$ for $N * N$ points. However, when 2D DFT computation is mapped as a distributed computing tasks on WSNs, the key consideration is shifted from computation complexity to communication cost.

In this chapter, an efficient NDFT algorithm is proposed for implementing the in-network Fourier analysis in WSNs on the HDACS model shown in the chapter 3 aimed at minimizing communication overhead, reducing execution time, and improving power efficiency. We exploit CS to realize a highly efficient NDFT algorithm in WSNs.

The salient steps of the CS-based NDFT are as follows:

1. Divide the sensing field into a hierarchy of clusters.
2. Collect data from randomly placed sensors at their corresponding cluster heads.
3. Perform the interpolation step of NDFT algorithm to obtain the expanded uniform data, and perform 2D Block FFT inside each cluster head.
4. At each cluster head, take sparse random measurements of the Fouler coefficients using CS. and then forward them to the cluster heads in the next level.
5. After receiving the data, perform CS recovery algorithm and 2D block FFT algorithm inside each cluster head to obtain the Fourier coefficients in this level.
6. Repeat step 4 and step 5 until the nodes at the highest level of the cluster hierarchy have the final Fourier coefficients for the entire data field.

Using theoretical analysis as well as simulations on a Java based SIDnet-SWANS sensor network simulating platform, we demonstrate that the proposed method has significant advantages over the previous work (54), in terms of execution time, energy consumption, signal-to-noise ratio (SNR), and communication overhead.

4.2 Related Work

For Fourier analysis in randomly deployed wireless sensor networks, NDFT has been considered a suitable tool to obtain frequency components of spatial irregularly sampled data. However, direct computation of NDFT requires the use of all samples. It is a energy-consuming task due to complex local computations and global communication patterns. In our previous work (54), we have addressed this problem by presenting a novel hybrid structure consisting of local interpolation step of NDFT algorithm and a new global FFT formulation to optimize the data communication pattern and therefore improve the power efficiency. For the sake of completeness, the main idea in (54) is summarized below.

4.2.1 NDFT algorithm

The basic idea behind computing Fourier coefficients of NDFT is to find n equispaced data based on M given nonequispaced data (56), where $n > M$. For a d dimensional space, let $\prod^d := [-\frac{1}{2}, \frac{1}{2}]^d$, $I_N := \{(N\prod)^d \bigcap \mathbb{Z}^d\}$, $I_M := \{(M\prod)^d \bigcap \mathbb{Z}^d\}$, set $\mathbf{V}$ denotes the time or spatial location and set $\mathbf{X}$ represents the frequency location. For nonequispaced data samples $f(v_{\mathbf{j}}), \text{where } v_{\mathbf{j}} \in \mathbf{V}, \mathbf{j} \in I_M$. Algorithm 6 shows fast computation of Fourier coefficients of M nonuniform data given in the time or spatial domain when $d = 1$.

4.2.2 Global Separable 2D FFT Formulation

N points FFT computation needs $K = \log(N)$ phases. Assume each sensor node is represented with a binary index $< b_K b_{K-1} .... b_1 >$. Two operations has been defined in one phase. In phase i, butterfly communication and computation are implemented as the first operation,

Algorithm 6 NDFT algorithm

Step 1. Choose window function $\tilde{\psi}(x)$, where $x \in \mathbf{X}$. Precompute its Fourier series $C_k(\tilde{\psi})$, where $k \in I_N$. (When it comes to the d dimension case, the window function will be $\tilde{\psi}(\mathbf{x}) = \prod_{t=1}^{d} \tilde{\psi}(x_t)$ for $\mathbf{x} \in \mathbb{R}^d$ instead.)

Step 2. Set up

$$g_l := \sum_{l=v_j n-m}^{v_j n+m} f(v_j)\tilde{\psi}(v_j - \frac{l}{n})$$

where m is the size of window function.

Step 3. Compute N points DFT of g_l

$$\hat{g}_k := \sum_{l=-n/2}^{n/2} g_l e^{-2\pi i k l/n}$$

Step 4. Compute

$$\tilde{h}_k := \frac{\hat{g}_k}{nC_k(\tilde{\psi})}$$

$\tilde{h}_k$ is the approximate frequency component of the data field.

which is the same as traditional FFT computation. In the second operation, binary index shuffling function has been defined and works on the last $i+1$ bits of binary index representation. The sensors exchange data according to the change of index in each phase, so that every sensor contains the data of newly shuffled binary index representation. Algorithm 7 shows the binary index shuffling function which enables all the butterfly computation only between two physical neighboring sensors. This property shortens transmission distance and thereby reduces transmission power. For additional details we refer to (54)

4.3 Proposed Power-efficient NDFT Implementation Design

The power-efficient NDFT implementation investigated is based on a novel compressive sensing (CS) data aggregation architecture, which consists of a hierarchy of clusters. The CS data aggregation architecture adopts the compressive sensing method as a compression tool to

Algorithm 7 The binary index shuffling function

if Phase $i < K/2$ **then**
Circular left shift the rightmost $i + 1$ bits of binary index one bit from $< b_K b_{K-1} \cdots b_{K/2+1} b_{K/2} \cdots b_{j+1} b_1 \cdots b_{j-1} b_j >$ to $< b_K b_{K-1} \cdots b_{K/2+1} b_{K/2} \cdots b_1 b_2 \cdots b_j b_{j+1} >$.
else if $i = K/2$ **then**
Binary index is flipped entirely from $< b_K b_{K-1} \cdots b_{K/2+1} b_1 b_2 \cdots b_{K/2-1} b_{K/2} >$ to $< b_{K/2} b_{K/2-1} \cdots b_2 b_1 b_{K/2+1} \cdots b_{K-1} b_K >$.
else if $i > K/2$ **then**
Circular left shift one bit from $(K/2+1)^{th}$ bit to $(i - K/2 + 1)^{th}$ bit of binary index and keep the rightmost $K/2$ bits unchanged, i.e., binary index changes from $< b_{K/2} b_{K/2-1} \cdots b_j b_1 \cdots b_{j-2} b_{j-1} b_{K/2+1} \cdots b_{K-1} b_K >$ to $< b_{K/2} b_{K/2-1} \cdots b_1 b_2 \cdots b_{j-1} b_j b_{K/2+1} \cdots b_{K-1} b_K >$.
end if

reduce the amount of data transmitted, and therefore achieves power efficiency. The following subsections introduce the main techniques involved in the NDFT task.

4.3.1 Initial Data Reduction Operation

The interpolation step of the NDFT algorithm expands the sampled data and represents it on a regular uniform grid to compute $N \times N$ global FFT. The traditional method extracts the first $N \times N$ data samples, removes the remaining data and then performs the FFT computation. This method is not directly applicable in sensor network, since it inevitably causes unbalanced sensing task distribution among clusters and renders data sensed from some clusters useless. In order to solve this problem we adopt a new data truncation method. Since the redundant data is distributed evenly over all clusters, we remove the same fraction of them within each cluster to shrink the data size.

4.3.2 Initial Data Shuffling Operation

The decimation-in-time (DIT) radix-l FFT algorithm is applied in the following global FFT computation task, which requires data reorganization before starting butterfly computation.

The new data groups have been set up based on the same remainder when their indexes are divided by radix l along row-wise and column-wise directions. We find it convenient to introduce the notation $J_N = \{0, 1, \cdots, N-1\}$, where N is a positive integer.

Suppose a $N \times N$ nodes network with $A \times A$ data in each cluster can be divided as $L \times L$ clusters, where $L = N/A$. Assume unit transmission distance between any neighboring clusters. For any data with index $(x, y) \in J_N^2$, the transmission distance for each data is $|x\%L - \lfloor x/A \rfloor| + |y\%L - \lfloor y/A \rfloor|$, where $\%$ is modulo operator and $\lfloor \quad \rfloor$ is floor operator.

4.3.3 Global Computation Operation

After cluster heads finish the aforementioned operations in the first level, DFT is calculated to obtain the initial Fourier coefficients. Random measurements are taken for CS and transmitted to their parent cluster heads at the next level. For consistency as model set in (58), we adopt the same 2D uniformly random network deployment and use the same parameters and notations in the following analysis.

In a N-sensor hierarchical structure with T levels, suppose $N_i^{(l)}$ is the number of sensors in cluster l at level i, n is the number consisting of one parent cluster head and its children cluster heads in two consecutive levels, $M_i^{(l)}$ is the measurements after performing CS or the amount of data for transmission. Signal sparsity factor of data field is assumed as K. In level $i (i \geq 2)$, each cluster head has $n-1$ children nodes. (Note: n, N are defined as model parameters different from previous notation.) The following operations are performed sequentially at each cluster head.

(a) Receive $(n-1)M_{i-1}$ measurements from its children nodes.

(b) Perform CS recovery algorithm to get Fourier coefficients F_i for each children node.

(c) Calculate DFT according to the following 2D block DIT radix-l FFT algorithm with $n * F_i$ Fourier coefficients to obtain Fourier coefficients F_{i+1}.

(d) Use CS to compress the data size of F_{i+1} into random measurements M_{i+1} and then send them to its parent cluster head C_{i+1}.

2D Block decimation-in-time radix-l FFT theorem. *2D FFT formula can be expressed in the following way:*

$$F(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X[m,n] e^{-j2\pi(\frac{mu}{M}+\frac{nv}{N})1}$$

Let

$$X = \begin{bmatrix} X_{00} & X_{01} & \cdots & X_{0(l-1)} \\ X_{10} & X_{11} & \cdots & X_{1(l-1)} \\ \vdots & & \ddots & \vdots \\ X_{(l-1)0} & & \cdots & X_{(l-1)(l-1)} \end{bmatrix}$$

---

[1] *m, n, M, N are only dummy variables in the theorem.*

Where $X_{qp} = X[lm+q, ln+p]$ *for* $m \in J_M, n \in J_N$ *and* $q, p \in J_l$. *Assume* $\hat{X}_{qp}$ *is the* $M * N$ *points Fourier transform of sequence* X_{qp}.

$$\begin{aligned}
F(u,v) &= \sum_{m=0}^{lM-1}\sum_{n=0}^{lN-1} X[m,n]e^{-j2\pi(\frac{mu}{lM}+\frac{nv}{lN})} \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} X[lm,ln]e^{-j2\pi(\frac{lmu}{lM}+\frac{lnv}{lN})} \\
&+ X[lm,ln+1]e^{-j2\pi(\frac{lmu}{lM}+\frac{(ln+1)v}{lN})} + \cdots \\
&+ X[lm+1,ln]e^{-j2\pi(\frac{(lm+1)u}{lM}+\frac{lnv}{lN})} + \cdots \\
&+ X[lm+1,ln+1]e^{-j2\pi(\frac{(lm+1)u}{lM}+\frac{(ln+1)v}{lN})} + \cdots \\
&+ X[lm+l-1,ln+l-1]e^{-j2\pi(\frac{(lm+l-1)u}{lM}+\frac{(ln+l-1)v}{lN})} \\
&= \hat{X}_{00} + \hat{X}_{01}e^{-j2\pi\frac{v}{lN}} + \cdots \\
&+ \hat{X}_{10}e^{-j2\pi\frac{u}{lM}} + \cdots + \hat{X}_{11}e^{-j2\pi(\frac{u}{lM}+\frac{v}{lN})} + \cdots \\
&+ \hat{X}_{(l-1)(l-1)}e^{-j2\pi(\frac{(l-1)u}{lM}+\frac{(l-1)v}{lN})}
\end{aligned}$$

It can also be expressed as:

$$F(u,v) = \sum_{q=0}^{l-1}\sum_{p=0}^{l-1} \hat{X}_{qp}e^{-j2\pi(\frac{qu}{lM}+\frac{pv}{lN})} \tag{4.3.1}$$

where $u \in J_M, v \in J_N$. *For* $a, b \in J_l$,

$$F(u+aM, v+bN) = e^{-j2\pi(\frac{a}{l}+\frac{b}{l})}F(u,v) \tag{4.3.2}$$

Equation (Equation 4.3.1) and (Equation 4.3.2) are utilized to achieve the Fourier coefficients in each level. 2D block FFT algorithm guarantees the sparse signal representation, and therefore enables the feasibility of CS. Fewer and fewer nodes participate in the task with the increasing levels, but Fourier coefficient size grows exponentially at the same time. CS reduces the amount of data for transmission to grow on a logarithmic scale. We will show in the fol-

lowing section that for the global FFT algorithm, 2D block FFT algorithm implemented on CS data aggregation architecture works much better than the existing state of the art.

4.4 Comparison of Related Work and Current Work

Since the local interpolation step of the NDFT algorithm is same as in previous work (54) in the initial phase, we mainly focus on the energy consumption with respect to global FFT algorithm. Previous work initialized the data shuffling operation to shorten transmission distance and therefore improved power efficiency. The data aggregation mechanism followed separable 2D FFT algorithm. The method in this work implements DFT task on multi-scale data aggregation architecture, and adopts a 2D block FFT algorithm to obtain the Fourier coefficients. Therefore, power efficiency comparison is essentially the comparison of two types of 2D FFT algorithms.

4.4.1 Theoretical Analysis

Energy efficiency has been evaluated theoretically from the perspective of amount of data transmitted and associated energy consumption.

4.4.1.1 Data transmitted

Let us assume that after performing the local interpolation step of the NDFT algorithm, cluster heads at level one have the uniform data with identical size M_u. We get $N_i^{(l)} = n^{i-1}M_u, M_i^{(l)} = K\log N_i^{(l)} = K\log\left(n^{i-1}M_u\right)$. The amount of data M for the whole task is:

$$\begin{aligned} M &= \sum_{l=1}^{|C_1|}(N_1^{(l)}-1)+\sum_{i=2}^{T}\sum_{l=1}^{|C_i|}(n-1)M_{i-1}^{(l)} \\ &= N-n^{T-1}+(n-1)n^{T-1}K(\log nS_1+\log M_uS_2) \end{aligned}$$

where $S_1 = \sum_{i=1}^{T-1} \frac{i}{n^i} = \frac{n^{-1}(1-n^{-(T-1)})}{(1-1/n)^2} - \frac{T-1}{n^T(1-1/n)}$ and $S_2 = \sum_{i=1}^{T-1} \frac{1}{n^i} = \frac{n^{-1}(1-n^{-(T-1)})}{1-1/n}$. We get an exact energy consumption value rather than an approximate range because the interpolation step of NDFT algorithm converts irregular sampling into a uniform one.

4.4.1.2 Energy consumption

We assume that the cost of transmission of a single bit over a distance $E_i^{(l)}$ is a function of transmission distance and data size. $E_i^{(l)}$ is modeled as $E_i^{(l)} = c_s + \sum_{j \in v_i^{(l)}} c(d_i^{(j)})^\alpha M_i^{(l)}$, where $v_i^{(l)}$ is the collection of children nodes, c_s is a constant startup energy consumption for each data transmission task, c is a constant transmission cost for unit data size per unit distance, and α is the power loss exponent. The number of c, c_s and α depend on the hardware and algorithms for various application tasks.

In a large dense uniformly and randomly distributed sensor network, $\sum_{j \in v_i^{(l)}} (d_i^{(j)})^\alpha \approx 4\frac{(n-1)}{s_i} \int_0^{b_i} \int_0^{b_i} (x^2+y^2)^{\alpha/2} \mathrm{d}x\mathrm{d}y$, where $b_i = \frac{1}{2} s_i^{1/2}$ when $i \geq 2$. And $\sum_{j \in v_i^{(l)}} (d_i^{(j)})^\alpha \approx 4\frac{(N_1^{(l)}-1)}{s_i} \int_0^{b_i} \int_0^{b_i} (x^2+y^2)^{\alpha/2} \mathrm{d}x\mathrm{d}y$ when $i = 1$.

Therefore, the total transmission energy cost is:

$$
\begin{aligned}
E &= \sum_{l=1}^{|C_1|} c_s + A_1(N_1^{(l)} - 1) \\
&+ \sum_{i=2}^{T} \sum_{l=1}^{|C_i|} c_s + A_1 M_i^{(l)} (n-1) n^{(i-1)(\alpha-1)/2} \\
&= n^{T-1} c_s (1 + S_2) + A_1 (N - n^{T-1}) \\
&+ A_1 K (n-1) n^{T-1} [S_3 \log n + S_4 \log M_u]
\end{aligned}
$$

Where we define $A_1 = \frac{c}{1+\alpha} 2^{-\alpha} \pi s^{(\alpha-1)/2}$, $S_3 = \sum_{i=1}^{T-1} i n^{i(\alpha-3)/2} = \frac{n^{(\alpha-3)/2}(1-n^{(\alpha-3)(T-1)/2})}{(1-n^{(\alpha-3)/2})^2} - \frac{(T-1)n^{(\alpha-3)T/2}}{1-n^{(\alpha-3)/2}}$ and $S_4 = \sum_{i=1}^{T-1} n^{i(\alpha-3)/2} = \frac{n^{(\alpha-3)/2}(1-n^{(\alpha-3)(T-1)/2})}{1-n^{(\alpha-3)/2}}$.

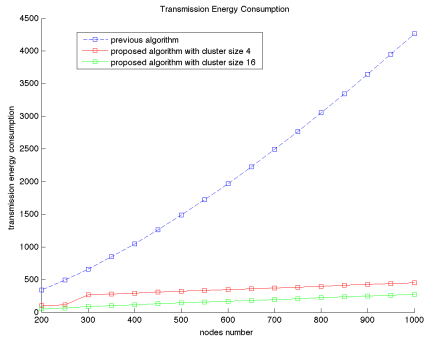


Figure 19. Transmission Energy Consumption Comparison

Data size M_u for each transmission is constant in paper (54). Therefore, transmission distance pays a key role for energy consumption. In the case of 1D seperarable row and column FFT, $H_{1D} = N_1^{3/2} - N_1 - 1/2 N_1 \log N_1 + PC(N_1)$ for N_1 nodes. If $N = N_1^2$ nodes constitute a 2D network, $H_{1D} = N^{3/4} - N^{1/2} - 1/4 N^{1/2} \log N + PC(\sqrt{N})$. Therefore, transmission distance for the whole data aggregation task is $H_{2D} = 2N_1 H_{1D} = 2N^{5/4} - 2N - 1/2 N \log N + 2N^{1/2} PC(\sqrt{N})$ and its corresponding energy consumption is evaluated as $E_{2D} = c_s + cd^{\alpha} M_u H_{2D}$, where d is unit distance between two neighbor finest clusters.

Figure 19 shows the quantitive results of the energy consumption, where constant parameters c_s, c, K, s are assumed as unity and α as 2. The proposed NDFT implementation method requires much less energy for transmission. The advantage is much more obvious with the increase of network size. Besides, more power efficiency will also be achieved by increasing the cluster size. The theoretical result gives a simple energy consumption comparison without considering the external inferences from the realistic network, such as packet collision, network

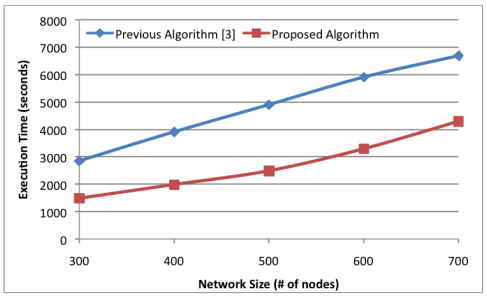


Figure 20. Simulated Execution Time Comparison

travel congestion, synchronism, etc. In the following section, we investigate performance based on implementations on a simulation platform.

4.4.2 Performance Evaluation

The simulation of the NDFT task is conducted on SIDnet-SWANS (16), a simulator and integrated development platform for sensor networks of various exploratory-design applications. We tested multiple network sizes from 300 to 700 nodes. The field size is fixed as $4000 * 4000 m^2$, and average nodes distribution density increases from $18.75/\text{km}^2$ to $43.75/\text{km}^2$. The cluster size n is set as 4, the communication system follows IEEE 802.15.4/4a standards for low data rate Wireless Personal Data Networks, and the data transmission rate is set as $40000 bps$.

The following results show the advantage of current work with respect to execution time for the whole task, distribution of transmission energy consumption, SNR evaluation, and communication overhead with respect to packet collision.

1. Execution time

Compared with computation time, communication time is the dominant execution time for the whole task. We compare the execution time starting from the point where cluster heads finish local interpolation step of NDFT algorithm so as to remove the interferences from different startup overheads for building up the connectivity of networks and different network hierarchy construction mechanisms. Figure 20 shows the comparison of execution time of previous algorithm and the proposed NDFT implementation on CS data aggregation architecture in terms of global FFT computation. The proposed algorithm shortens the execution period by almost 50%. The main reasons that lead to significantly reduced execution time are explained as follows: During the process of practical simulation, separable 2D global FFT algorithm applied in paper (54) introduces extra communication overhead. This operation is to solve the problem that arises for the instance when the network size is approximate 300 to 800 nodes, in which case at least 16*16 nodes are required to participate in the global FFT computation. The initial 8*8 clusters in the first level cannot provide enough nodes for global computation. We solved this problem in (54) by dividing the space into 2*2 subspaces to include more nodes. But this operation inevitably introduces the backward data aggregation and increases the global execution time. Another important reason is that the data shuffling operation aimed at reducing the butterfly communication distance also brings the extra time delay. Since this delay is in each phase, it explains the phenomenon that it takes at almost two times of execution time for the whole task. However, these two problems don't exist in the multi-scale hierarchical data gathering architecture.

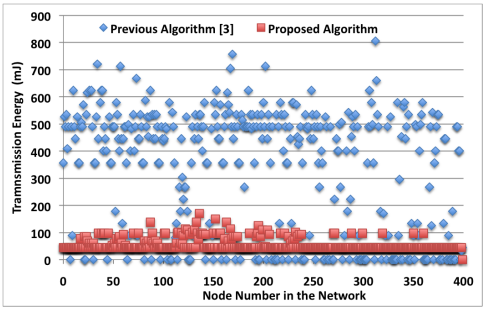


Figure 21. Distribution of transmission energy consumption comparison for 400 nodes with cluster size 4

2. Energy consumption distribution for transmission

   The comparison of the distribution of energy consumption for transmission among 400 sensors is given in Figure 21. The result verifies the proposed method outperforms the previous work significantly. There are some reasons which explain this great power efficiency improvement. In the previous algorithm (54), more nodes are required for global FFT computation. For computing 16*16 points FFT in a 400 sensor network, 256 nodes are required for row-wise and column-wise FFT computation. However, the task mainly rests on the cluster heads at higher levels in the hierarchical data gathering architecture. Leaf nodes at the first level are more than the cluster heads. After forwarding the sensed data to the cluster heads, they are in idle mode. On the other hand, CS applied as compression method reduces the amount of data size for transmission, and therefore its transmission cost is low.

3. SNR evaluation

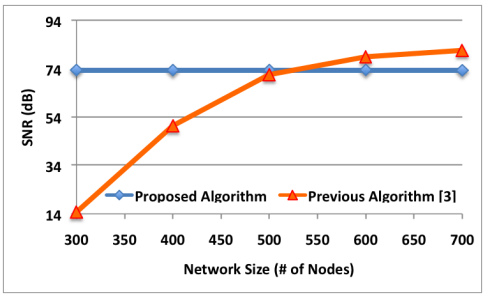


Figure 22. Signal-to-noise ratio results

We compare the Fourier coefficients computed using the proposed distributed algorithm with Fourier coefficients computed using a centralized NDFT algorithm in the sink node where all the data is available. Figure 22 shows the SNR results for the constant data field with uniform noise. The proposed algorithm preserves the accuracy regardless of the network size.

Our simulation results shows that SNR is relatively independent of the network size. Therefore, for communication efficiency this observation favors global FFT step on smaller size networks. We can improve the SNR performance if the local interpolation step considers neighboring clusters, which has been showed in previous work (54).

4. Communication overhead: packet collision

Additional advantage of the proposed algorithm is that it reduces the packet collision phenomenon substantially. If data aggregation is mainly performed on a linear array (1D) topology, it is inevitable that the two neighboring sensors will send their data concurrently and cause packet collision. Previous work (54) has this problem and it is especially severe

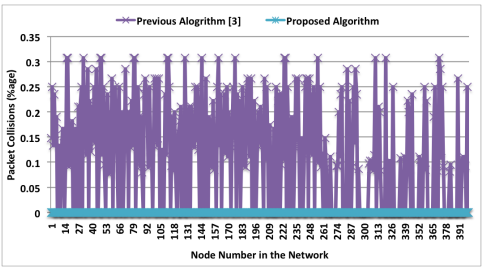


Figure 23. Packet collision ratio comparison for 400 nodes

in the middle stage when it involves a large number of nodes to send data over a long distance. Completely avoiding the collision is not easy by designing scheduling algorithm in the application layer. But this problem can be mitigated in the hierarchical data structure to a great extent. Clustering mechanism enables all the transmission only within the restricted clustering area. Since the cluster heads generate the data from their children nodes, it is easy to design the scheduling algorithm to arrange children nodes to transmit their data individually in a dedicated time slot. At the upper levels of the cluster hierarchy, increase in the spatial distance further reduces the possibility of collisions. This property also improves the power efficiency by reducing the retransmission rates. The simulation result in Figure 23 verifies the theoretical analysis.

CHAPTER 5

ADAPTIVE HIERARCHICAL DATA AGGREGATION USING COMPRESSIVE SENSING (A-HDACS) FOR NON-SMOOTH DATA FIELD

5.1 Introduction

In chapter 3, we improved CS-based data aggregation by proposing a Hierarchical Data Aggregation using Compressive Sensing (HDACS) (59) that introduced a hierarchy of clusters into CS data aggregation model and achieved significant energy efficiency. However, most of the previous works used CS under the assumption that data field is smooth with negligible white Gaussian noise. In these schemes, signal sparsity is calculated globally based on the entire data field. In more realistic scenarios, where data field may have regional fluctuations or it is piecewise smooth, existing CS based data aggregation schemes will yield poor compression efficiency. The sparsity constant K is usually a big number, with large probability, when the field consists of bursts or bumps. In such cases, the number of CS measurements $M = K \log N$ is bigger than N, where N is the local cluster size. In order to take full advantage of CS for its great compression capability, we propose an Adaptive Hierarchical Data Aggregation using Compressive Sensing (A-HDACS) scheme.The proposed scheme adaptively chooses sparsity values based on signal variations in local regions.

Our solution is based on the observation that the number of CS random measurements from any region (spatial or temporal) should correspond to the local sparsity of the data field, instead of global sparsity. Intuitively, it should work well because the nodes are more correlated with each other in a local area than the entire global area. Also, it is easy to compute the local sparsity, particularly when a data aggregation scheme is based on a hierarchical clustering scheme. Also, in order to compute global sparsity, *apriori* knowledge of the data field is required. We show that the proposed A-HDACS scheme enables more sensor nodes to utilize compressive sensing compared to the HDACS scheme (59) that employs global sparsity based compressive sensing. Two types of data fields: smooth data field with multiple Gaussian bumps and piecewise smooth data field are chosen for performance evaluations. Smooth data field with multiple Gaussian bumps represents some typical environment monitoring applications, such as radioactive waste treatment, air pollution monitoring, chemistry release monitoring, etc. (60). And piecewise smooth data field manifests a type characteristic of two flat measurement surfaces connected by a sharp segmentation in many applications, such as indoor VS. outdoor temperature monitoring. The effectiveness of the proposed scheme has been demonstrated by using SIDnet-SWANS (16) sensor simulation platform. For the smooth data field with multiple Gaussian bumps, A-HDACS reduces energy consumption by $\approx 6\%$ to 10%, depending on the network size. Similarly, for the piecewise smooth data field, it reduces energy consumption by $\approx 23.36\%$ to 30.17% depending on the network size. We observe higher gains in larger network sizes. The experimental results are consistent with our theoretical analysis.

TABLE V

PARAMETERS DEFINITION

N	The network size
T	The total level of the hierarchy
$N_i^{(l)}$	The cluster size at level i in cluster l
$M_i^{(l)}$	The amount of data transmitted after performing CS at level i in cluster l
C_i	The collection of clusters at level i
$\|C_i\|$	The number of cluster at level i in cluster l where $\|C_i\| = n^{T-i}$

5.2 Proposed Adaptive HDACS (A-HDACS) Scheme

The basic idea behind A-HDACS is that CS random measurements for each sensor that need to be communicated are determined by the sparsity of data field within each cluster at different levels of the data aggregation tree.

For consistency, we adopt the same notations as chapter 3 shown in (59), showed in Table V.

In order to capture varying sparsity of the data field based on local regions, we also define the following variables.

- K_T: the whole data field sparsity

- K_{i_T}: threshold defined as $K_{i\_T} = \max_{l \in C_i} \{ \frac{N_i^{(l)}}{\log N_i^{(l)}} \}$ at level i

- $K_i^{(l)}$: sparsity of the data field in cluster l at level i

Besides, we also define two types of nodes: CS-enabled nodes and CS-disable nodes. In CS-enabled nodes the data collected is large and sparse enough that CS pays off. On the other hand, in CS-disabled nodes the data collected is small and/or not sparse enough to yield the benefits of CS.

The salient steps of A-HDACS implemented on the multi-resolution data collection hierarchy are as follows:

1. At level one, each leaf node sends its sensed data to its cluster head without applying CS. The cluster head receives the data from all its member nodes and performs the conventional transformation to obtain the signal representation and its sparsity factor $K_1^{(l)}$. Then it compares $K_1^{(l)}$ to $\frac{N_1^{(l)}}{\log N_1^{(l)}}$. If $K_1^{(l)} < \frac{N_1^{(l)}}{\log N_1^{(l)}}$, it becomes a CS-enabled sensor and computes the CS random measurements. The amount of data that needs to be transmitted is $M_1^{(l)} = K_1^{(l)} \log N_1^{(l)}$; otherwise, it disables itself as a CS-disabled node and transmits $N_1^{(l)}$ data directly to its parent clusters.

2. At level i ($i \geq 2$), cluster head receives packets from its member nodes. If it receives CS random measurements, the CS recovery algorithm is performed to recover all the data. After cluster head gets all the data from the children nodes, it projects the whole data into a transform domain to obtain the signal representation and its sparsity factor $K_i^{(l)}$. If $K_i^{(l)} < \frac{N_i^{(l)}}{\log N_i^{(l)}}$, cluster head turns itself as a CS-enabled node and computes CS random measurements with length $M_i^{(l)} = K_i^{(l)} \log N_i^{(l)}$; otherwise, it becomes a CS-disabled node and sends raw data directly.

3. Repeat step 2) until cluster head at the top level, T, obtains and recovers the entire data.

5.3 Analysis of Data Field Sparsity

Proposition 1. *In HDACS, if $K_T > K_{i\_T}$, all the nodes at the level equal to and below i are all CS-disabled nodes.*

Proof. Define: $f(x) = \frac{x}{\log x}$. since $f'(x) = \frac{\log x - \frac{1}{\ln 2}}{(\log x)^2} > 0$ when $x > 3$. Therefore, $f(x)$ is a monotonous increasing function when $x > 3$.

1. At level i, if $K_T > K_{i\_T}$ then $K_T > \frac{N_i^{(l)}}{\log N_i^{(l)}}$. In HDACS, CS requires the amount of data to be transmitted $M_i^{(l)} = K_T \log N_i^{(l)}$. Therefore, $M_i^{(l)} > N_i^{(l)}$ for $\forall j \in C_i$. Thus clusters at level i are all CS-disabled nodes.

2. At level j and $j < i$, since $N_j^{(l)} < N_i^{(p)}$ for $\forall l \in C_j$ and $\forall p \in C_i$, $K_{i\_T} > K_{j\_T}$. So $K_T > K_{j\_T} > \frac{N_j^{(l)}}{\log N_j^{(l)}}$ and $M_j^{(l)} = K_T \log N_j^{(l)} > N_j^{(l)}$. Thus the nodes at levels below i are also all CS-disabled nodes.

□

On the other hand, if $\exists l \in C_i$ s.t. $K_T > K_{i\_T} > \frac{N_i^{(l)}}{\log N_i^{(l)}} > K_i^{(l)}$ at level i. In A-HDACS, since $M_i^{(l)} = K_i^{(l)} \log N_i^{(l)} < N_i^{(l)}$, CS can be utilized.

Let's define C_i' a set consisting of all clusters whose cluster heads are CS-disabled nodes at level i, and ρ_i CS-disabled clusters percentage. In cluster l, $\sigma_i^{(l)}$ is defined as the percentage of the CS-disabled children clusters in a CS-disabled cluster at level i, where $\sigma_i^{(l)} \in \{\frac{1}{n}, \frac{2}{n}, \cdots, \frac{n}{n}\}$. We get $\rho_i = \frac{|C_i'|}{|C_i|}$ at level i; and $\rho_{i-1} = \frac{\sum_{l=1}^{|C_i'|} n\sigma_i^{(l)}}{|C_{i-1}|}$ at level $i-1$.

Proposition 2. *If $K_T > K_{i\_T}$, the CS-disabled nodes of A-HDACS at the level equal to and below i are only ζ percentage of that of HDACS and $\zeta < 1$.*

Proof. Let's define $\sigma_i = \frac{1}{|C_i'|}\sum_{l=1}^{|C_i'|}\sigma_i^{(l)}$, which shows the average ratio of CS-disabled children clusters to their parent clusters. Therefore, we get $\rho_{i-1} = \frac{n|C_i'|\sigma_i}{|C_{i-1}|} = \frac{|C_i'|\sigma_i}{|C_i|} = \rho_i\sigma_i$. Follow the same derivation, $\rho_{i-2} = \rho_i\sigma_i\sigma_{i-1}, \rho_{i-3} = \rho_i\sigma_i\sigma_{i-1}\sigma_{i-2}, \cdots, \rho_1 = \rho_i\sigma_i\sigma_{i-1}\cdots\sigma_2$. In summary, the ratio of CS-disabled clusters in HDACS at level i and below level i is:

$$\zeta = \frac{\sum_{j=1}^{i}|C_j|\rho_j}{\sum_{j=1}^{i}|C_j|} = \frac{\sum_{j=1}^{i}|C_j|\rho_i(\sigma_i\sigma_{i-1}\cdots\sigma_{j+1})}{\sum_{j=1}^{i}|C_j|}$$

Since ρ_i and σ_i are equal to or less than 1, ζ is strictly less than 1. Therefore, it proves that only ζ percent of the nodes at the level equal to and below i are CS-disabled nodes for A-HDACS. □

When it comes to the level higher than i, i.e. $i < t < T$, the results are diversified. We summarize them as follows:

1. If $\frac{N_t^{(l)}}{\log N_t^{(l)}} > K_t^{(l)} > K_T$, CS is enabled in both HDACS and A-HDACS. HDACS requires fewer measurements than A-HDACS. But it comes another problem that whether or not HDACS can guarantee the recovery accuracy when a local area has significantly more data variations compared to the global area.
2. If $K_t^{(l)} > \frac{N_t^{(l)}}{\log N_t^{(l)}} > K_T$, HDACS enables CS but A-HDACS not. But it also has the same problem as scenario 1).
3. If $K_T > \frac{N_t^{(l)}}{\log N_t^{(l)}} > K_t^{(l)}$, A-HDACS enables CS but HDACS does not.
4. If $\frac{N_t^{(l)}}{\log N_t^{(l)}} > K_T > K_t^{(l)}$, both HDACS and A-HDACS enable CS. A-HDACS requires fewer measurements than that in HDACS.

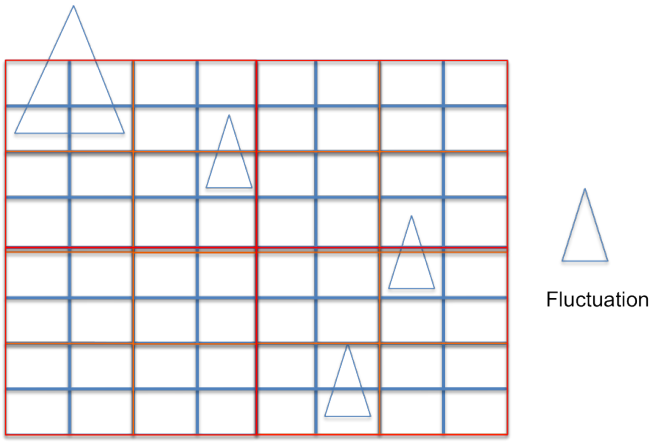
5. In the remaining conditions, CS is disabled for both schemes.

To better understand this analysis, Figure 24 gives an example of a smooth data field with a few local fluctuations and its corresponding logical tree in both HDACS and A-HDACS. In Figure 24(b), the local variations causes the large global sparsity K_T, and therefore leads to plenty of nodes to be classified as CS-disabled nodes in HDACS. In the meanwhile, since in A-HDACS sparsity constants K_i are derived based on local variations in each cluster i, there is bigger percentage of CS-enabled nodes in A-HDACS than that in HDACS.

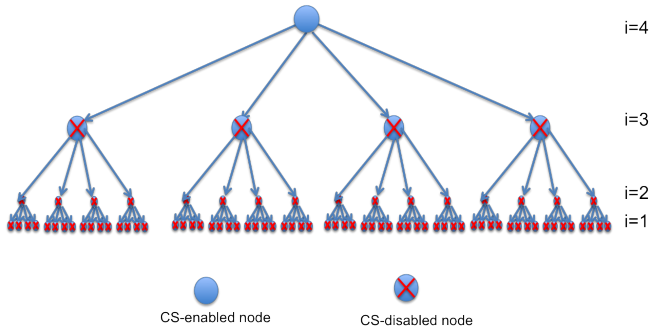
5.4 Performance Evaluation

5.4.1 Simulation Settings

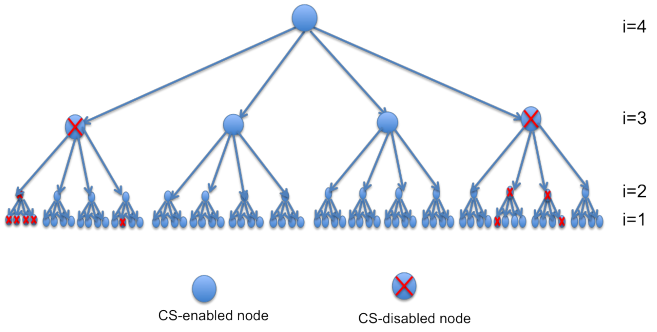
We evaluate the performance of the proposed A-HDACS scheme using SIDnet-SWANS (16), a JAVA based sensor network simulation platform. In our experiments we tested multiple network sizes, ranging from 300 to 800 sensor nodes, populated in a fixed field size of $4000 * 4000m^2$ area. The average nodes distribution density varies from $18.75/\text{km}^2$ to $50/\text{km}^2$. Additionally, the communication system follows IEEE 802.15.4/4a standards for low data rate wireless personal data networks. In the simulation, the data transmission rate is set as 40000bps, and the length of a standard message for transmission is 133 bytes (88 bytes for the payload of interest from application layer and 45 bytes for side control information from other layers of the network stack). Theoretically, it takes $133 * 8/40000 = 26.6$ms for one message per transmission per hop. Furthermore, we neglect the noise interference from the channel and ignore the possibility of random packet loss. For power consumption parameters, it follows Mica2 Motes specs, where the radio transmission cost is our major concern. The default unit cost of radio transmission



(a) A smooth data field with fluctuations



(b) HDACS logical tree



(c) A-HDACS logical tree

Figure 24. An example of a smooth data field with fluctuations and its corresponding logical tree in HDACS and A-HDACS

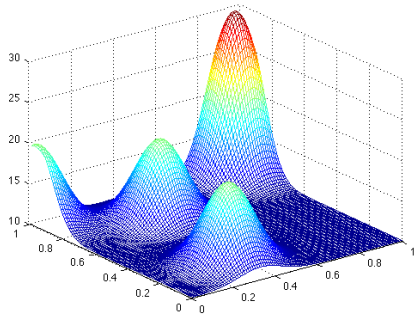
is $81\mu J/ms$ for one message. We use Table I for the major parameters assignments in the simulations.

Figure 25(a) shows a constant data field filled with randomly located Gaussian bumps. Its maximum height is 10 units and decays with exponential rate of 0.01. Figure 25(b) depicts a smooth data field with a discontinuity along the line $x = y$, where the readings from smooth area are either 10 or 20 plus independent Gaussian noise with zero mean and 0.01 variance.

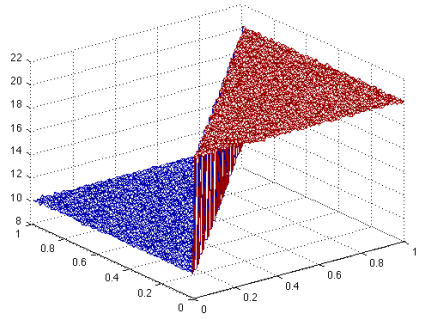
Besides, we make use of Discrete Cosine Transform (DCT) to capture the sparsity of data field. DCT is a suboptimal transformation for sparse signal representation and approaches the ideal optimal transform when the correlation coefficient between adjacent data elements approaches unity (61). Figure 25(c) and Figure 25(d) plot the coefficients distributions when two data fields are projected into DCT space. As we can see, only a few coefficients with large magnitudes capture the most signal energy and the rest coefficients decay rapidly.

5.4.2 The Nodes Distribution

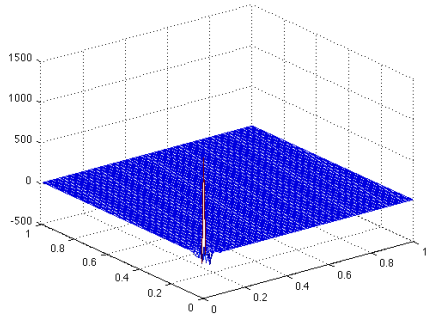
Figure 26 shows the SIDnet simulation results of A-HDACS and HDACS for two types of data fields with network size 400, where black nodes denote CS-enabled nodes, gray nodes denote CS-disabled nodes, and white nodes are the leaf nodes at level one. The scattered fluctuations in data field with Gaussian bumps cause less percentage of CS-enabled nodes shown in Figure 26(a) than that in piecewise smooth data field shown in Figure 26(b). In piecewise data field, CS-disabled nodes are mainly placed around the discontinuity of two flat surfaces in the line $x = y$. And the clusters away from this line can fully utilize CS. Figure 26(c) and Figure 26(d) depict the node distributions for both data fields in HDACS. And almost no CS can be



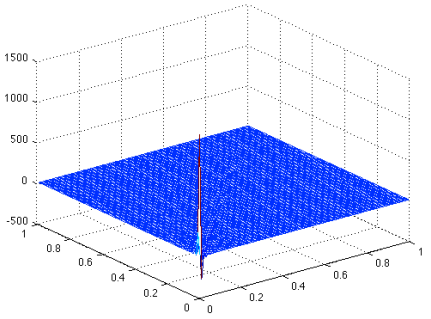
(a) Smooth data field filled with Gaussian bumps



(b) Piecewise data field



(c) DCT domain of smooth data field filled with Gaussian bumps



(d) DCT domain of piecewise data field

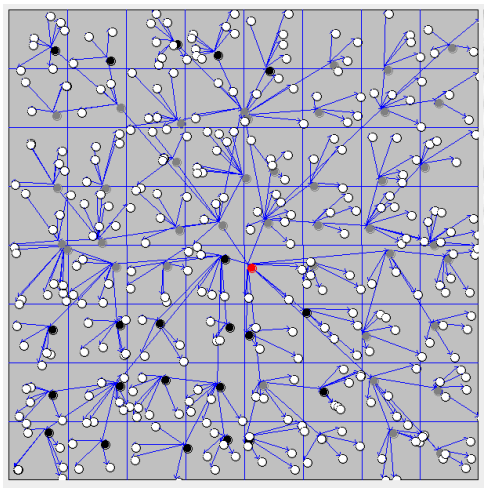
Figure 25. Data Fields and their corresponding DCT Domain

performed at the lower level except a few nodes at the top levels. It demonstrates the significant improvement of CS usage efficiency in A-HDACS and it is consistent with theoretical analysis in Section 5.3.

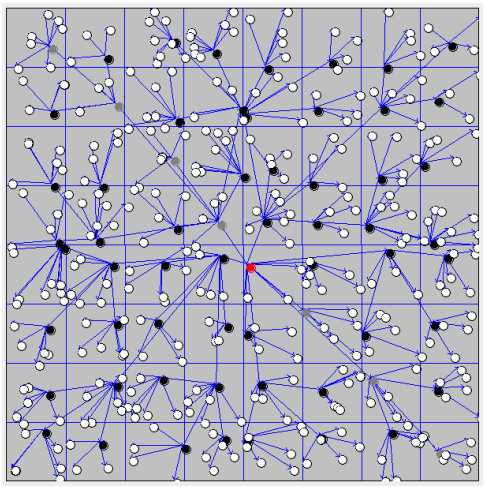
5.4.3 Data Recovery Fidelity

Common signals are usually K-compressive – K entries with significant magnitudes and the other entries rapidly decaying to zero. We perform signal truncation process to get K-sparse signal. In the simulation, we tested different signal truncation thresholds so as to get as many CS-enabled nodes as possible without compromising signal recovery fidelity. The percentage of the first dominant magnitude is set up as truncation threshold.

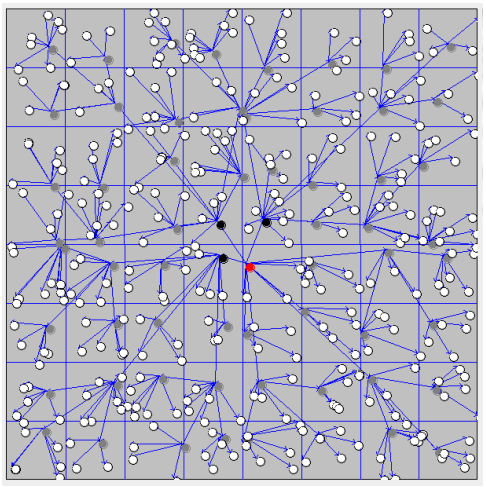
In the evaluation, Mean Square Error (MSE) of recovered signal in the root node (sink) is defined as the average difference between recovered data and actual reading values for all the sensors. Figure 27 depicts MSE versus DCT truncation threshold for two types of data field with network size 400. Since small truncation threshold filters out fewer significant entries than larger thresholds, it reduces the overall average square error. Figure 27 shows that MSE of the smooth data field with Gaussian bumps is below 0.066 when DCT threshold is smaller than 0.0225, and it increases dramatically when DCT threshold becomes larger. In the case of the smooth data field with Gaussian bumps, fluctuations in the signal cause the increase in the number of DCT coefficients that have significant magnitudes, therefore truncation process becomes less effective. Relatively, piecewise field has more smooth clustering area with only a few significant entries. Its MSE is controlled under a negligible range when DCT threshold is in the $[0.005, 0.03]$ interval.



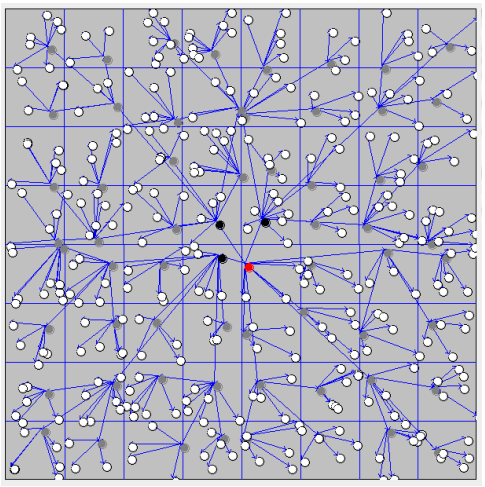
(a) A-HDACS: smooth data field filled with Gaussian bumps



(b) A-HDACS: piecewise data field



(c) HDACS: smooth data field filled with Gaussian bumps



(d) HDACS: piecewise data field

Figure 26. The SIDnet simulation results of A-HDACS and HDACS with network size 400: black nodes denote CS-enabled nodes, gray nodes denote CS-disabled nodes, white nodes are the leaf nodes at level one, and red node denotes the sink.

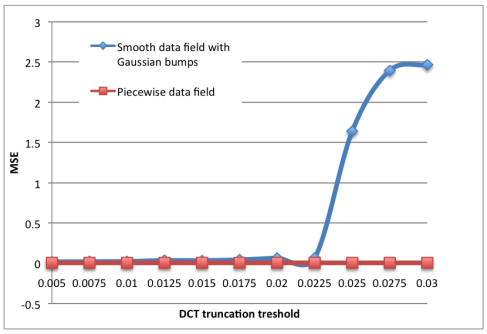
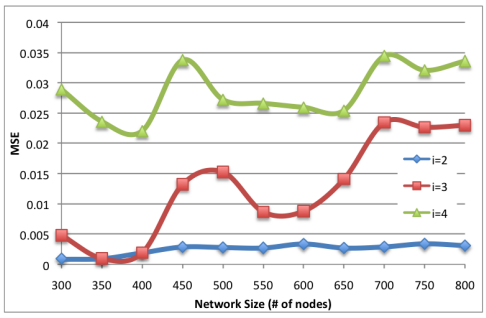
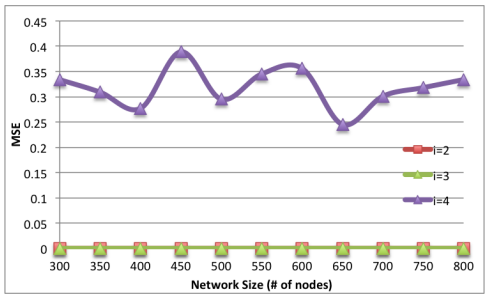


Figure 27. MSE versus DCT truncation threshold with network size 400

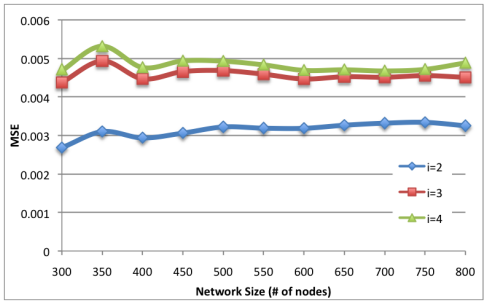
In the simulation results presented here onwards, DCT magnitudes bigger than 1% of the first dominant coefficient are preserved. Figure 28 shows data recovery results for different data aggregation schemes measured under two data fields with different levels. In both data fields, MSE results deteriorate with increase in the number of levels in the data collection tree, owing to the errors propagation of the signal truncation in the hierarchy. In the meantime, although HDACS scheme outperforms AC-HDACS at the lower level for both data fields, its data recovery results are significantly worse than those of A-HDACS, when it comes to the top level. It is due to the fact that raw data transmission without processing ensures zero data distortion but it fails to make a good use of CS for a broad area owning to local data fluctuations. It further proves HDACS cannot be applied to non-smooth because it sacrifices the data recovery fidelity. Moreover, comparing Figure 28(a) with Figure 28(c), overall piecewise data field has smaller errors than the smooth data field with Gaussian bumps. It is due to relatively fewer scattered fluctuations in the piecewise smooth data field.



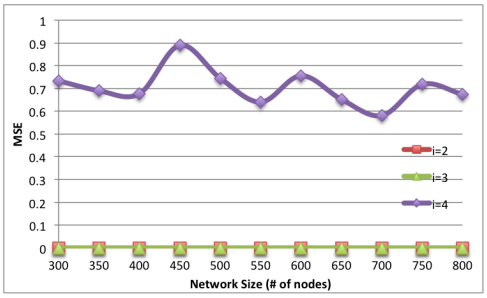
(a) A-HDACS: smooth data field filled with Gaussian bumps



(b) HDACS: smooth data field filled with Gaussian bumps

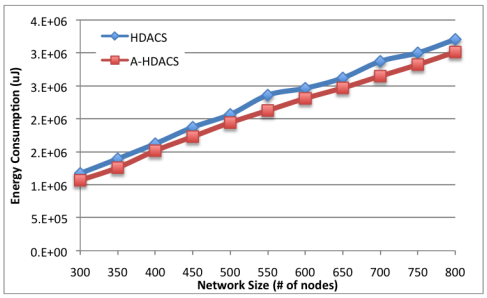


(c) A-HDACS: piecewise data field

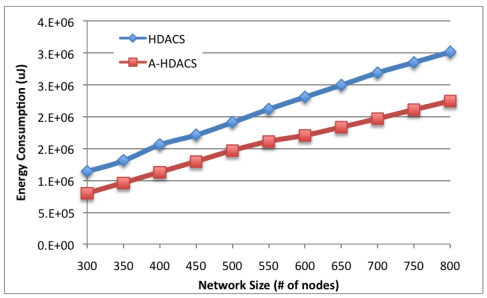


(d) HDACS: piecewise data field

Figure 28. Data recovery mean square error (MSE) results



(a) Smooth data field filled with Gaussian bumps



(b) Piecewise data field

Figure 29. Total Transmission Energy Cost versus Different Network Sizes

5.4.4 Energy Consumption

Figure 29(a) and Figure 29(b) show energy consumption versus networks size for two types of data field. A-HDACS consumes only 90.1% $\sim$ 94.20% energy of HDACS for different network sizes. Although plenty of fluctuations in the data field affects A-HDACS to apply CS to a certain degree, it still captures the sparsity feature within a few cluster area. But HDACS is insensitive to the local area. When the data field slightly change, it loses its data compression capability. This advantage is obvious, when it comes to the piecewise data field. Figure 29(b) shows that A-HDACS can save around 23.36% $\sim$ 30.17% battery power, compared to HDACS. The results demonstrate that significant energy efficiency can be obtained by the proposed technique.

CHAPTER 6

SPATIO-TEMPORAL HIERARCHICAL DATA AGGREGATION USING COMPRESSIVE SENSING (ST-HDACS)

6.1 Introduction

Considering the real-world applications especially for environment monitoring, data collection is usually periodically performed over an extended time period (34). The temporal data model is a natural topic of interest for CS-based data aggregation schemes. Distributed Compressive Sensing (DCS) (62) exploits both intra- and inter-signal sparsity to lower sampling rate for each sensor, which purely relates to compressing the data volume in temporal domain. The application-oriented method in (63) refers to the use of spatio-temporal compressive sensing but its focus is primarily on how to obtain a network feature metric rather than data aggregation. They refer to the idea of CS to investigate the sparsity of the data field to justify reduction of samples but actually do not implement the CS technique in their work.

Motivated by all the previous data gathering schemes aimed at reducing data redundancy in the spatial or temporal domain, we investigate an energy-efficient CS-based data collection scheme that targets to reduce spatial as well temporal redundancy using an integrated framework with focus on reducing the cost of data transmission. We refer to the approach as: Spatio-Temporal Hierarchical Data Aggregation using Compressive Sensing (ST-HDACS). The main idea of ST-HDACS consists of three key components:

- Firstly, for each instance of data collection, only a random subset of sensors in the network is selected to participate in the data aggregation.
- Secondly, our earlier work on Adaptive Hierarchical Data Aggregation using Compressive Sensing (A-HDACS) scheme (64) is incorporated for data transmission along the routing paths in the aggregation tree.
- Lastly, Matrix Completion (MC) (65) approach at the fusion node is adopted to recover all the data instances for the entire network sensed during the data collection time.

Since the MC problem tends to require complex recovery algorithm, extensive studies (65) (63) (66) have offered feasible and effective solutions. We believe it is a suitable choice for large data recovery with guaranteed accuracy.

In summary, the advantages of ST-HDACS are multifold:

- The fact that only a randomly selected subset of nodes representing the whole network participate in any given communication round saves the battery power and prolongs the network life time;
- For any given instance of data collection, the use of A-HDACS scheme removes the spatial data redundancy and reduces the transmission time.
- Furthermore, ST-HDACS is also well suited for practical applications. It offers an immediate answer to user query at any time with latest data. At any particular time t_n, if the data query is about some unselected sensors for which the fusion center has no immediate data, it can still answer the query by interpolation from nearest spatial and/or temporal

data. On the other hand, for a data query over the entire network for all the collections, Matrix Completion (MC) provides an accurate and effective solution for the large data recovery problem.

6.2 Spatio-Temporal Hierarchical Data Aggregation using Compressive Sensing

In this section, the details of ST-HDACS schemes are elaborated. Our goal is to reduce the amount of the data that is necessary for transmission under a guaranteed data recovery accuracy. The main idea is that at each data collection instant, a subset of randomly selected nodes implement A-HDACS and collect the corresponding subset of the data; in the end, the fusion center utilizes Matrix Completion (MC) algorithm to recover all the data for the entire network over the whole data collection period.

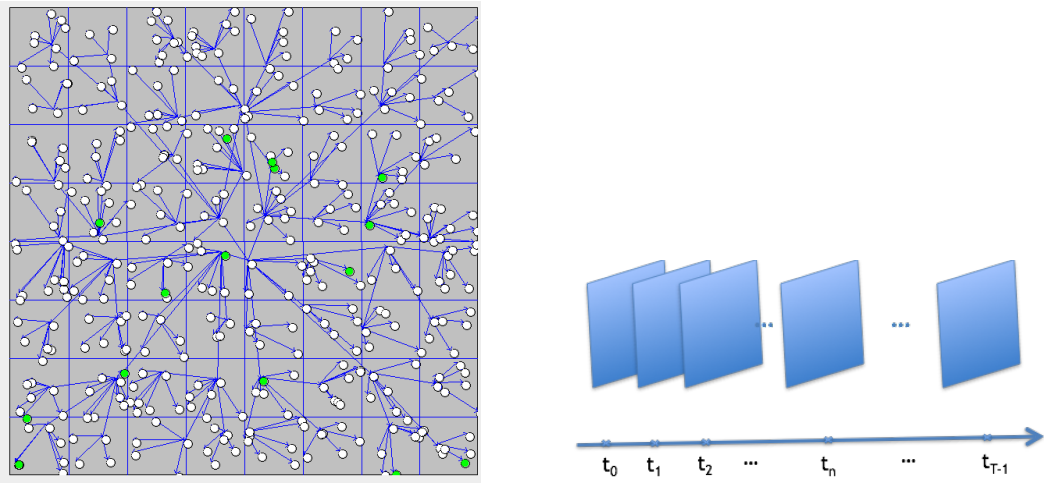
6.2.1 Problem Formulation

In ST-HDACS, at any time t_n, the network is configured into a hierarchy consisting of multi-level clusters of sensors and the data collection follows the same routing model as HDACS. Figure 30(a) illustrates the data propagation path for a single-instant data collection. Figure 30(b) shows a conceptual data model of interest, where each square represents one snapshot of observed data field.

6.2.1.1 Single-instant Data Collection

For a network consisting of N nodes, the cluster head at the top level (sometimes referred as fusion center) in HDACS-based schemes collects the CS random measurements as:

$$Y = \Phi\Psi\boldsymbol{\alpha} \tag{6.2.1}$$



(a) Data Field Snapshot

(b) Conceptual Temporal Data: Each Square Representing One Snapshot of Observed Data Field

Figure 30. Data Model

where Φ is a $M * N$ sensing matrix, $M = K \log N$ and K is the sparsity of data representation on the entire field, Ψ is a $N \times N$ transformation basis which sparsifies the data field and $\boldsymbol{\alpha}$ is a $N \times 1$ vector with entries representing data collected by each sensor.

6.2.1.2 Data Collections Over A Period

Over a time period $\{t_0, t_1, t_2, \ldots, t_{T-1}\}$, where T is an integer and $T \geq 1$, the fusion center collects the data:

$$\begin{aligned} \tilde{Y} &= \left[Y_{t_0}\ Y_{t_1}\ Y_{t_2}\ \ldots\ Y_{t_{T-1}} \right] \\ &= \Phi\Psi \left[\boldsymbol{\alpha}_{t_0}\ \boldsymbol{\alpha}_{t_1}\ \boldsymbol{\alpha}_{t_2}\ \ldots\ \boldsymbol{\alpha}_{t_{T-1}} \right]. \end{aligned}$$

Therefore, CS random measurements collected in the fusion center over a period are:

$$\tilde{Y} = \Phi\Psi\tilde{\boldsymbol{\alpha}},$$

where $\tilde{Y} \in \mathbb{R}^M \times \mathbb{R}^T$, $M = K \log N$ and $\tilde{\boldsymbol{\alpha}} = \left[\boldsymbol{\alpha}_{t_0}\ \boldsymbol{\alpha}_{t_1}\ \boldsymbol{\alpha}_{t_2}\ \ldots\ \boldsymbol{\alpha}_{t_{T-1}}\right] \in \mathbb{R}^N \times \mathbb{R}^T$.

In addition, we specify $\mathcal{A}(.)$ as a $N \times T$ matrix $\mathcal{Q}$, where

$$\mathcal{Q}(i,j) = \begin{cases} 0 & \text{if the entry is missing,} \\ 1 & \text{otherwise.} \end{cases}.$$

In ST-HDACS scheme, we apply $\mathcal{A}(.)$ operator into $\tilde{\boldsymbol{\alpha}}$ to represent the nodes which do not participate in this data collection. It implies that

$$\tilde{Y}' = \Phi\Psi\mathcal{A}(\tilde{\boldsymbol{\alpha}}).$$

Or put another way, $\mathcal{A}(\tilde{\boldsymbol{\alpha}}) = \mathcal{Q}. * \tilde{\boldsymbol{\alpha}}$, where $.*$ is an element-wise product.

Note that $\mathcal{A}(\tilde{Y})$ is not equal to $\Phi\Psi\mathcal{A}(\tilde{\boldsymbol{\alpha}})$, as the dimensionality of $\tilde{Y}$ and of $\tilde{\boldsymbol{\alpha}}$ are different. And $\mathcal{A}(.)$ or $\mathcal{Q}$ contains the network topology information: the entries 1 in each column of $\mathcal{Q}$ represent the selected nodes and the entries 0 for nonselected nodes at each data collection.

Consider $N_\lambda \in \mathbb{Z}$, such that $1 \leq N_\lambda \leq N$. Define $\lambda = N_\lambda/N$, which represents the percentage of nodes selected for each data collection. We define $\dot{\mathcal{A}}^{(\lambda)}(.)$ as a condensed version of $\mathcal{A}(.)$, which removes all nonzero entries. To be more specific, $\dot{\mathcal{A}}^{(\lambda)}(.)$ condenses the operand

into a $N_\lambda \times T$ matrix. In summary, the measurements collected at the fusion center over a period in ST-HDACS is defined as:

$$\tilde{Y}^{(\lambda)} = \Phi^{(\lambda)}\Psi^{(\lambda)}\dot{\mathcal{A}}^{(\lambda)}(\tilde{\boldsymbol{\alpha}}),$$

where $\tilde{Y}^{(\lambda)} \in \mathbb{R}^{\lceil K \log N_\lambda \rceil} \times \mathbb{R}^T$, $\Phi^{(\lambda)} \in \mathbb{R}^{\lceil K \log N_\lambda \rceil} \times \mathbb{R}^{N_\lambda}$ and $\Psi^{(\lambda)} \in \mathbb{R}^{N_\lambda} \times \mathbb{R}^{N_\lambda}$.

It is shown in (65) that if m entries are selected uniformly at random from a matrix M, where m obeys

$$m \geq Cn^{1.2} r \log n$$

for some positive numerical constant C and $n = \max\{n_1, n_2\}$ for $n_1 * n_2$ matrices of rank r, then with very high probability, they can be perfectly recovered by solving a simple convex optimization problem. In this work, we assume the network size is larger than the data collection time, that is $N > T$. Therefore, in order to guarantee the signal recovery fidelity, the condition

$$\lambda = \frac{N_\lambda}{N} = \frac{N_\lambda \times T}{N \times T} \geq \frac{CN^{1.2} r \log N}{N \times T} = CN^{0.2} r \log N/T$$

has to be satisfied. (In some other cases, when the data collection number is larger than the network size, the condition $\lambda \geq CT^{0.2} r \log T/N$ has to be satisfied and it does not affect the results in our case.)

6.2.2 Data Recovery

Data recovery involves two algorithms: Compressive Sensing (CS) and Matrix Completion (MC). In ST-HDACS, CS recovery algorithm is implemented to get a subset of data field immediately after each data collection. In the end, MC is utilized in the fusion center to recover the data for the entire network over the whole period.

6.2.2.1 Instantaneous Compressive Sensing Recovery

For instantaneous Compressive Sensing (CS) recovery, we re-visit Equation 6.2.1. We aim to solve:

$$\begin{aligned} &\text{minimize} \quad \| \Psi\boldsymbol{\alpha} \|_{l_0} \\ &\text{subject to} \quad Y = \Phi\Psi\boldsymbol{\alpha}, \end{aligned}$$

where we use l_1 norm to replace l_0 norm (67) (38) (39). Among numerous CS recovery algorithms (49) (50), a convex optimization-based approach — CoSaMP algorithm (43), utilizes the proxy (43) $y = \Psi * \Psi\boldsymbol{\alpha}$, which preserves the energy of K largest components of signal $\boldsymbol{\alpha}$, to approximate the K-sparse signal $\boldsymbol{\alpha}$. It has been proved to have fewer iteration steps than the other Orthogonal Matching Pursuit (OMP) algorithms, which we will exploit and utilize in our scheme.

Repeating this procedure iteratively for each data collection instant, we can recover the data $\dot{\mathcal{A}}^{(\lambda)}(\tilde{\boldsymbol{\alpha}})$ from CS measurements $\tilde{Y}^{(\lambda)}$. By adopting the structural information from $\mathcal{A}(.)$,

we can retrieve $\tilde{\boldsymbol{\alpha}}$, in which zero entries show the missing data that we are going to recover as follows.

6.2.2.2 Latent Matrix Completion Recovery

As the matrix obtained from the previous step contains only a subset of data with entries missing, that is: $\mathcal{A}(\tilde{\boldsymbol{\alpha}})$. We let $\tilde{X} = \mathcal{A}(\tilde{\boldsymbol{\alpha}})$ and strive to find a low-rank solution for the following optimization problem:

$$\begin{aligned} &\text{minimize} \quad rank(\tilde{\boldsymbol{\alpha}}) \\ &\text{subject to} \quad \mathcal{A}(\tilde{\boldsymbol{\alpha}}) = \tilde{X} \end{aligned}$$

We first decompose $\tilde{\boldsymbol{\alpha}}$ into three elements, i.e. $\tilde{\boldsymbol{\alpha}} = U\Sigma V^T$, where U is a $N \times N$ unitary matrix and V is a $T \times T$ unitary matrix. Σ is a $N \times T$ diagonal matrix containing the singular values. We can the factorize matrix $\tilde{\boldsymbol{\alpha}} = U\Sigma V^T = LR^T$, where $L = U\Sigma^{1/2}$ and $R = V\Sigma^{1/2}$. We seek L and R with the smallest sum of their Frobenius norms:

$$\begin{aligned} &\text{minimize} \quad \| L \|_F^2 + \| R \|_F^2 \\ &\text{subject to} \quad \mathcal{A}(LR^T) = \tilde{X} \end{aligned}$$

In practice, matrix $\tilde{\boldsymbol{\alpha}}$ may not be exactly low-rank, and the data to be recovered may contain errors itself. We therefore solve the following optimization problem instead:

$$\text{minimize} \quad \| \mathcal{A}(LR^T) - \tilde{X} \|_F^2 + \| L \|_F^2 + \| R \|_F^2$$

This is a convex optimization problem and many research efforts (65) (63) (66) have been made to offer effective solutions. We briefly introduce one that balances the computational complexity and the accuracies of results. A method of using alternative projection was introduced in (63). At the beginning of the procedure, randomly initialize L and R. Fix L as a constant and optimize R with linear least square method and then fix R as a constant and optimize L with least square method. After a moderate number of iterations, it will converge to the optimal solution for matrix L and R.

6.3 Performance Evaluation

6.3.1 Data Field

We use the real-world data as well as synthetic data to test the effectiveness and robustness of our approach. For the real-world data, we use the data from the National Data Buoy Center (http://tao.ndbc.noaa.gov/tao/data_download/search_map.shtml). We collected the daily sea surface temperature data measured by 52 nodes across the Pacific Ocean from 165 E to 95 W, 8N to 8S from December 21 to December 31, 2014, which serves as an underlying data field for surveillance. Figure 31 illustrates the sea surface temperature field on Dec 21, Dec 26, Dec 31. As we observe, each data field is globally smooth with a few boundaries marked by abrupt changes in values. And during this period, the data field does not have significant change temporally.

Considering the limitation of the scale from the real-world data, we also tested our scheme on two types of synthetic data fields: a smooth data field with discontinuity along the line $x = y$ and a smooth data field with randomly placed Gaussian bumps. For the piecewise constant

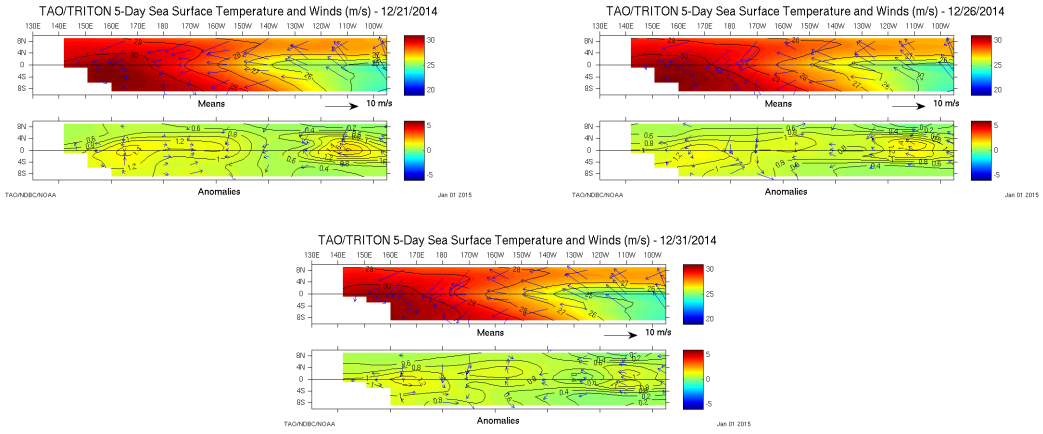
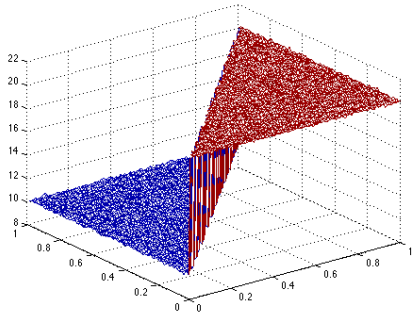


Figure 31. The Sea Surface Temperature Data Field

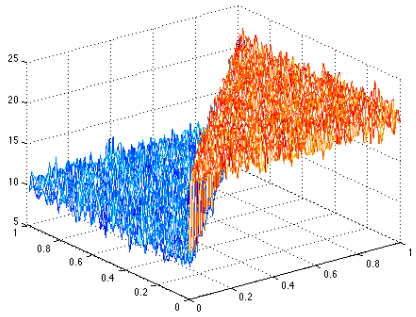
data field, the values are a constant number corrupted by identical independent Gaussian noise and the data field evolves in a way that the Gaussian noise starts from the standard deviation $\sigma = 0.1$ in the first data collection phase to $\sigma = 1$ at the last phase. And for the smooth data field with Gaussian bumps, we set up the initial exponential rate $\sigma = 1/48$ and the last one as $\sigma = 1/24$. Figure 32 shows these two types of the data fields and at the starting phase and the last phase.

6.3.2 Simulation Settings

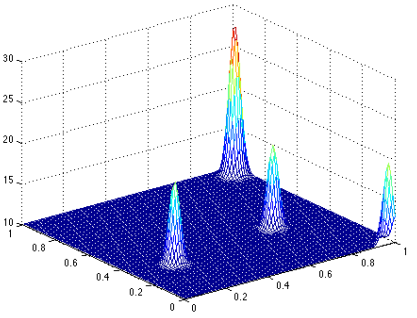
We implement the data gathering schemes on SIDnet-SWANS (16). In the network configuration settings, we deployed 300 to 800 sensors with 50 nodes as an increment each time. Each simulation contains ten data collection iterations, i.e. $T = 10$. For the real-world data set simulation, we use the ten days sea surface temperature data for each data collection. The data measured by each sensor are simulated by spatial interpolation. For the synthetic data, the



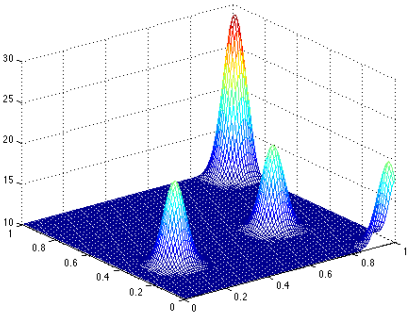
(a) Piecewise Constant Data Field Corrupted by Gaussian Noise with Initial $\sigma = 0.1$



(b) Piecewise Constant Data Field Corrupted by Gaussian Noise with Final $\sigma = 1$



(c) Smooth Data Field with Gaussian Bumps with Initial $\sigma = 1/48$



(d) Smooth Data Field with Gaussian Bumps with Final $\sigma = 1/24$

Figure 32. Synthetic Data Fields

data grid has higher resolution. At data collection instant t_n, we simulate the value of the data field as the temporal interpolation from the initial data field and the final one. Furthermore, in the initial phase, all the nodes are set up as active and participate in the data collection so as to construct the hierarchy. In the following nine iterations, λ percent of sensor nodes will be selected at each data collection. Furthermore, in order to keep the integrity of the hierarchy, only leaf nodes have rights to decide for themselves whether or not to sleep. We obtain this goal by generating a random number which follows the uniform distribution. If the random number is larger than λ, the leaf node is set to sleep. A suitable choice of λ is also very critical for the data collection accuracy. In the simulation, we choose the values of λ as 0.25, 0.50, 0.75 and 1 respectively.

6.3.3 Simulation Results

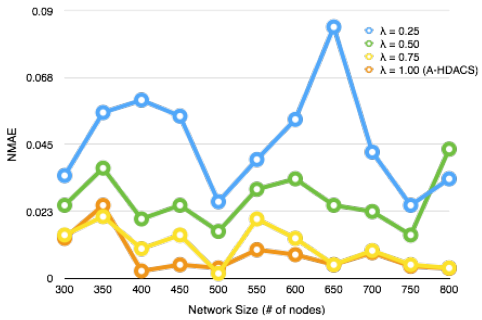
6.3.3.1 Data Collection Fidelity

We measure the fidelity of the data collected in the fusion center using the Normalized Mean Absolute Error (NMAE)(63):

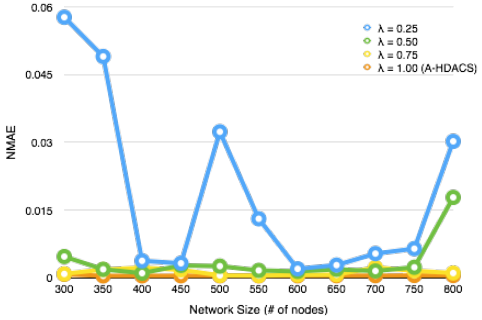
$$NMAE = \frac{\sum_{i=0}^{T-1} |\alpha_{t_i} - \hat{\alpha}_{t_i}|}{\sum_{i=0}^{T-1} |\alpha_{t_i}|},$$

where α_{t_i} is a $N \times 1$ data vector sensed by all the sensors at data collection t_i and $\hat{\alpha}_{t_i}$ is the final data stored at the fusion center.

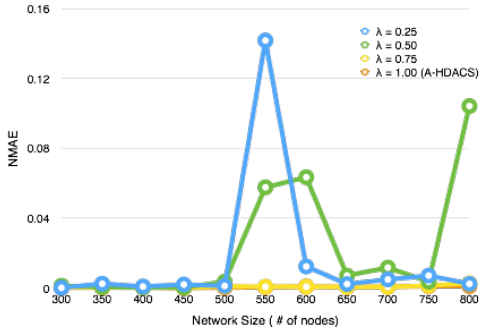
Figure 33 indicates the data collection fidelity evaluated by NMAE for the real-world data and two synthetic data fields. Comparing Figure 33(a) with Figure 33(b) and Figure 33(c),



(a) Sea Surface Temperature Data



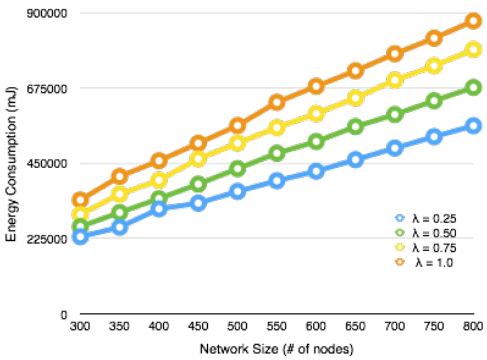
(b) Piecewise Constant Data Field Corrupted by Gaussian Noise



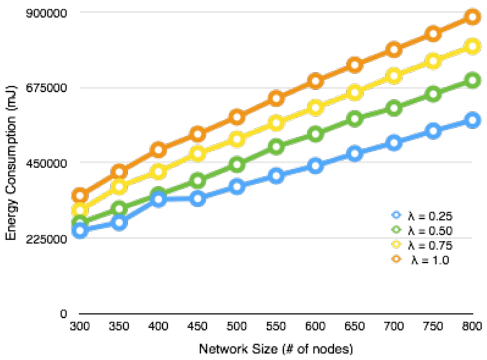
(c) Constant Data Field with Gaussian Bumps

Figure 33. Data Collection Fidelity

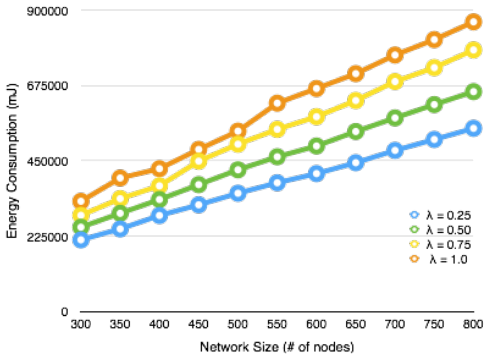
we observe that the result from real-world data set is less smooth than that of synthetic data fields, which is within our expectation. Also observe that all the figures show similar trends in terms of NMAE versus λ. When $\lambda = 1$, in other words, all the nodes are active for all the data collections, then the scheme is equivalent to independently implementing the A-HDACS scheme ten times. In this scenario, the minimum NMAE is achieved. As λ is decreased, more and more nodes are entitled to sleep and the data collection fidelity decreases accordingly. Note



(a) Sea Surface Temperature Data



(b) Piecewise Constant Data Field Corrupted by Gaussian Noise



(c) Constant Data Field with Gaussian Bumps

Figure 34. Total Amount of Energy Consumption versus Network Sizes For Different Data Fields

that when $\lambda = 0.75$, NMAEs are controlled in the limited range which the results are as good as the case when $\lambda = 1$. Although when $\lambda < 0.75$, we observed a few results in the simulations where the NMAE is large, Figure 33 demonstrates the data recovery accuracy is still controlled within a certain acceptable range which is independent of network sizes.

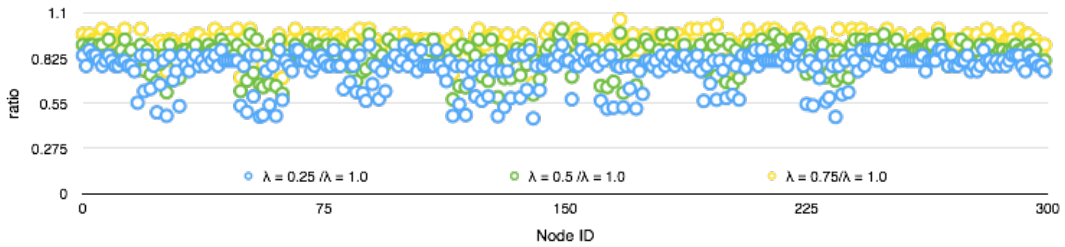


Figure 35. The Energy Consumption Ratio of Different Choices of $\lambda < 1$ over $\lambda = 1$ for Each Node in a 300-node Networks

6.3.3.2 Energy Consumption

Figure 34 plots the trends in the total amount of energy consumed versus the network size for the real world datasets as well as the synthetic data fields. We notice that there is no significant difference in the total energy consumption for different data fields, which implies that the contribution due to the variation inherent in the data field is relatively small when it comes to multiple data collections over a period. In the meantime, all figures show that with the decrease of λ, the amount of total energy consumption drops drastically. The results coincide with our expectation that smaller percentage of the nodes selected for each data collection improve the energy efficiency effectively. In the case of $\lambda = 1$, it is equivalent to the regular A-HDACS scheme applied successively, which further demonstrates the superiority of ST-HDACS scheme compared to repeated A-HDACS at different time instants for the case of the multiple data collections in WSNs.

Figure 35 shows the distribution of energy consumption ratio of $\lambda < 1$ over $\lambda = 1$ for each node in terms of network size 300. Since different data field types show the same distribution

pattern, we only show one figure here. It is noteworthy that in this implementation, when $\lambda = 0.25$, node energy consumption saving is up to 54.3% ; when $\lambda = 0.5$, the energy saving for some nodes is uppermost 42.7%; when $\lambda = 0.75$, the maximum energy saving for some nodes is 32.9%. The result manifests the energy efficiency of ST-HDACS scheme for individual nodes.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this dissertation, we have investigated distributed computing and data processing tasks on the WSNs and in the meanwhile proposed novel algorithms to obtain the power efficiency.

Implementing data processing task in a distributed fashion by factoring in energy efficiency is an important problem on wireless sensor networks. In chapter 2 we presented a solution to realize Fourier analysis in randomly distributed networks through implementing Nonuniform DFT (NDFT) algorithm. We proposed a hybrid structure consisting of local clustering interpolation step of NDFT and global FFT implementation. Interpolation step of NDFT within clusters avoids long-haul distance data transmission without compromising the fidelity of expanded uniformed data. In the meantime, Power-efficient FFT (PE-FFT) is formulated by introducing data shuffling at each stage of FFT to save energy cost with respect to global data communication. From the theoretical analysis and computer simulation in SIDnet-SWANS platform, we demonstrate that our method improve energy consumption efficiency effectively.

In chapter 3, we have investigated a novel power-aware data collection scheme — Hierarchical Data Aggregation using Compressive Sensing (HDACS) for large scale dense wireless sensor networks. We addressed this problem by incorporating Compressive Sensing (CS) in a multi-level data aggregation hierarchy. Cluster heads at different levels encode the propagated data as

CS random measurements, which substantially reduce data volume for transmission and therefore improve energy efficiency. We proved theoretically the advantages of the proposed HDACS over the existing work, in terms of the amount of data communicated in each cluster head, the total data volume for transmission, data compression ratio. Furthermore, we constructed a novel energy model by taking the computation cost in processor into account and we showed that computation cost is insignificant compared to communication cost with the aid of the real hardware specs. In order to demonstrate its effectiveness and feasibility, we have implemented different CS-based data aggregation schemes using the real datasets as well as synthetic data on SIDnet-SWANS sensor simulation platform to evaluate the performance. The signal recovery results measured by SNR and energy consumption have validated the superiority of the HDACS schemes over the existing state of the art.

Another novel power-efficient implementation of nonuniform 2D Fourier analysis over wireless sensor network has been presented in chapter 4. In this work, we implemented the task on the HDACS model showed in chapter 3. We adopt a multi-resolution strategy to realize a local interpolation step of NDFT algorithm at the first level followed by a global 2D block FFT algorithm at the high levels. In contrast to previous work in chapter 2, the proposed method shows significant advantages from theoretical analysis as well as in simulation results in terms of execution time, transmission power efficiency, SNR and packet collision phenomenon.

In chapter 5, adaptive data aggregation scheme referred to as Adaptive Hierarchical Data Aggregation using Compressive Sensing (A-HDACS) has been presented to perform data aggregation in non-smooth multimodal data fields. Existing CS based data aggregation schemes for

WSNs are inefficient for such data fields, in terms of energy consumed and amount of data transmitted. The A-HDACS solution is based on computing sparsity coefficients using signal sparsity of data gathered within local clusters. We analytically prove that A-HDACS enables more clusters to use CS compared to conventional HDACS. The simulation evaluated on SINnet-SWANS also demonstrates the feasibility and robustness of A-HDACS and its significant improvement of energy efficiency as well as accurate data recovery results.

In Chapter 6, Spatio-Temporal Hierarchical Data Aggregation scheme using Compressive Sensing (ST-HDACS) has been developed to improve power efficiency for the data collection problem with multiple data collection phases. We addressed the limitation of the existing CS-based data aggregation schemes for WSNs by removing the spatial data redundancy in the routing path using A-HDACS and removing the temporal data redundancy by choosing a subset of sensor nodes at any given instance and using MC at the fusion node to interpolate the missing data. The simulation results demonstrate significant energy savings compared to repeated use of A-HDACS. It also shows that with a suitable choice of the subset size, the data fidelity is limited in an acceptable range.

7.2 Future Work

There are numerous open issues that will be further explored in the context of our work in progress.

- In the HDACS-based schemes, cluster size is predefined assuming ideal deployment settings. In real applications, it may not fit all different scenario of the sensor field. It's inevitable to offer a suitable solution to adapt to the observed environment. Crucial fac-

tors in the deployment of the networks, such as adaptive and non-uniform cluster sizes need to be further investigated.

- We have assumed that the sensor readings are perfect and reflect the data field accurately; however, the in real deployments, the sensor readings may be corrupted by noise. A good estimation method to recover the data and reflect the true information is desirable. Furthermore, its impact on the data gathering and data analysis in WSNs shall be explored.
- Since Compressive Sensing (CS) is the key factor in the proposed HDACS model, the recovery accuracy and complexity play an important role as well. Different CS recovery algorithms shall be investigated in the HDACS-based scheme. For example, Distributed Compressed Sensing (DCS) (62), that takes the data spatial correlation into account, seems to be a very promising CS recovery algorithm. Its impact on computation and communication efficiency in WSNs shall be investigated.
- Other distributed computing tasks, such as Discrete Wavelet Transform(DWT), can be explored to fully utilize the advantage of the HDACS model to improve power efficiency.

CITED LITERATURE

1. Abowd, G. D. and Mynatt, E. D.: Charting past, present, and future research in ubiquitous computing. ACM Trans. Comput.-Hum. Interact., 7(1):29–58, March 2000.

2. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E.: A Survey on Sensor Networks. IEEE Communication Magazine, pages 102–114, August 2002.

3. Culler, D., Estrin, D., and Srivastava, M.: Overview of Sensor Networks. IEEE Computer Society, August 2004.

4. Sinha and Amitabha: JouleTrack-a Web based tool for software energy profiling. Design Automation Conference, 2001. Proceedings, 2001.

5. Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H.: Energy-Efficient Communication Protocol for Wireless Microsensor Networks. Proceedings of the 33rd Hawaii International Conference on System Sciences, 2000.

6. Chiasserini, C. F. and Rao, R. R.: On the concept of distributed digital signal processing in wireless sensor networks. page 260. IEEE MILCOM, October 2002.

7. Canli, T., Cupta, A., and Khokhar, A.: Power Efficient Algorithm for Computing Fast Fourier Transform over Wireless Sensor Networks. pages 549–556. International Conference on Computer Systems and Applications, March 8 2006.

8. Potts, D., Steidl, G., and Tasche, M.: Fast Fourier transforms for nonequispaced data: A tutorial. Modern Sampling Theory Mathematics and Applications, pages 1–23, 2001.

9. Beylkin, G.: On the fast fourier transform of functions with singularities. Applied and Computational Harmonic Analysis, 2(4):363 – 381, 1995.

10. Brandt, A.: Multilevel computations of integral transforms and particle interactions with oscillatory kernels. Computer Physics Communications, 65(1–3):24 – 38, 1991.

11. Dutt, A. and Rokhlin, V.: Fast fourier transforms for nonequispaced data, {II}. Applied and Computational Harmonic Analysis, 2(1):85 – 100, 1995.

CITED LITERATURE (Continued)

12. Kunis, S.: Nonequispaced FFT - Generalisation and InversionT: Generalisation and Inversion. Shaker Verlag GmbH, Germany (February 9, 2007), 2007.

13. Dutt, A. and Rokhlin, V.: Fast Fourier transforms for nonequispaced data. volume 14, pages 1368–1393. SIAM J. Sci. Comput., 1993.

14. Dutt, A.: Fast Fourier Transform for Nonequispaced Data. SIAM Journal on Scientific Computing, 14(6):1368–1393, 1993.

15. Mitra, S.: Digital Signal Processing: A Computer-Based Approach. McGraw-Hill Science/Engineering/Math, 2010.

16. Ghica, O. C.: SIDnet-SWANS Manual. Technical report, Northwestern University, March 3 2010.

17. JiST / SWANS Java in Simulation Time / Scalable Wireless Ad hoc Network Simulator. http://jist.ece.cornell.edu.

18. Barr, R.: Jist-Java in Simulation Time User Guide. http://jist.ece.cornell.edu/docs.html.

19. Han, K., Liu, Y., and Luo, J.: Duty-cycle-aware minimum-energy multicasting in wireless sensor networks. In IEEE/ACM TRANSACTIONS ON NETWORKING, June 2013.

20. Subramanian, R. and Fekri, F.: Sleep scheduling and lifetime maximization in sensor networks: Fundamental limits and optimal solutions. In 5th ACM IPSN, number 218-225, 2006.

21. Han, K., Xiang, L., Luo, J., and Liu, Y.: Minimum-energy connected coverage in wireless sensor networks with omni-directional and directional features. In ACM MobiHoc, number 85-94, 2012.

22. Li, X.-Y., Song, W.-Z., and Wang, W.: A unified energy-efficient topology for unicast and broadcast. In ACM MobiCom, pages 1–15, 2005.

23. Luo, J. and Hubaux, J.-P.: Joint sink mobility and routing to increase the lifetime of wireless sensor networks: The case of constrained mobility. In IEEE/ACM TRANSACTIONS ON NETWORKING Networking, volume 18, pages 871–884, 2010.

CITED LITERATURE (Continued)

24. Xing, G., Wang, T., Jia, W., and Li, M.: Rendezvous design algorithms for wireless sensor networks with a mobile base station. In 9th ACM MobiHoc, pages 231–240, 2008.

25. Polastre, J., Hill, J., and Culler, D.: Versatile low power media access for wireless sensor networks. In Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems, SenSys '04, pages 95–107, New York, NY, USA, 2004. ACM.

26. Candes, E. and Wakin, M.: An Introduction To Compressive Sampling. Signal Processing Magazine, IEEE, 25(2):21–30, March 2008.

27. Baraniuk, R. G.: Compressive Sensing [lecture notes]. Signal Processing Magazine, IEEE, 24(4):118–121, 2007.

28. Romberg, J. and Wakin, M.: Compressed Sensing: A Tutorial. http://users.ece.gatech.edu/justin/ssp2007, August 26 2007.

29. Kong, L., Xia, M., Liu, X., Wu, M.-Y., and Liu, X.: Data loss and reconstruction in sensor networks. In IEEE INFOCOM, 2013.

30. Liu, G., Tan, R., Zhou, R., Xing, G., Song, W.-Z., and Lees, J. M.: Volcanic Earthquake Timing using Wireless Sensor Networks. In ACM/IEEE IPSN, 2013.

31. Roughan, M., Zhang, Y., Willinger, W., and Qiu, L.: Spatio-Temporal Compressive Sensing and Internet Traffic Matrices (Extended Version). Networking, IEEE/ACM Transactions on, 20(3):662–676, June 2012.

32. Cheng, J., Ye, Q., Jiang, H., Wang, D., and Wang, C.: An efficient data gathering algorithm based on matrix completion for wireless sensor networks. IEEE Transactions on Wireless Communications, 12(2), 2013.

33. Luo, C., Wu, F., Sun, J., and Chen, C. W.: Compressive Data Gathering for Large-Scale Wireless Sensor Networks. Beijing, China, September 20-25 2009. MobiCom.

34. Artiola, J., Pepper, I., and Brusseau, M.: Environmental Monitoring and Characterization. Elsevier Science, June 2004.

35. Luo, J., Xiang, L., and Rosenberg, C.: Does compressed sensing improve the throughput of Wireless Sensor Networks? Number 1-6, Cape Town, South Africa, May 2010. In Proceedings of the IEEE International Conference on Communications.

CITED LITERATURE (Continued)

36. Xiang, L., Luo, J., and Vasilakos, A. V.: Compressed Data Aggregation for Energy Efficient Wireless Sensor Networks. Number 46. the 8th IEEE SECON, 2011.

37. Chen, Y., Liestman, A., and Liu, J.: Energy-efficient data aggregation hierarchy for wireless sensor networks. In Quality of Service in Heterogeneous Wired/Wireless Networks, 2005. Second International Conference on, pages 7–7, August 2005.

38. Donoho, D. L.: Compressed Sensing. IEEE Trans. Inf. Theory, 52(4), 2006.

39. Romberg, J.: Imaging via Compressive Sampling. IEEE Signal Processing Magazine, March 2008.

40. Candes, E., Romberg, J., and Tao, T.: Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE Trans. Inf. Theory, 52(2):489–509, Feb 2006.

41. Candes, E.J., and Tao, T.: Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies? IEEE Trans. Inf. Theory, 52(12):5406–5425, Dec 2006.

42. Needell, Deanna, and Vershynin, R.: Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. Foundations of computational mathematics, 9(3):317–334, 2009.

43. Needell, D. and Tropp, J.: Cosamp: Iterative signal recovery from incomplete and inaccurate samples. Applied and Computational Harmonic Analysis, 26(3):301–321, May 2009.

44. Ciullo, Delia, Celik, G. D., and Modiano, E.: Minimizing transmission energy in sensor networks via trajectory control. pages 132–141. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on., Institute of Electrical and Electronics Engineers, 2010.

45. Shnayder, V., Hempstead, M., Chen, B.-r., Allen, G. W., and Welsh, M.: Simulating the power consumption of large-scale sensor network applications. In Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems, SenSys '04, pages 188–200, New York, NY, USA, 2004. ACM.

46. Atmel: http://www.atmel.com/images/doc2467.pdf. 2011.

CITED LITERATURE (Continued)

47. Baraniuk, R. G., Cevher, V., Duarte, M. F., and Hegde, C.: Model-Based Compressive Sensing. IEEE Trans. Inf. Theory, 56(4), April 2010.

48. Coifman, R., Geshwind, F., and Meyer, Y.: Noiselets. Applied and Computational Harmonic Analysis, 10:27–44, 2001.

49. Tropp, J. A. and Gilbert., A. C.: Signal recovery from random measurements via orthogonal matching pursuit orthogonal matching pursuit. IEEE Trans. Inf. Theory, pages 4655–4666, 2007.

50. Donoho, D. L., Tsaig, Y., Drori, I., and Starck, J.-L.: Sparse solution of underdetermined linear equations by stagewise Orthogonal Matching Pursuit (StOMP). IEEE Trans. Inf. Theory, 2007.

51. Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8 - Volume 8, HICSS '00, Washington, DC, USA, 2000. IEEE Computer Society.

52. Bandyopadhyay, S. and Coyle, E.: An energy efficient hierarchical clustering algorithm for wireless sensor networks. In INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, volume 3, pages 1713–1723, March 2003.

53. Ali, S. and Khokhar, A.: Distributed center location algorithm for fault-tolerant multicast in wide-area networks. In Reliable Distributed Systems, 1998. Proceedings. Seventeenth IEEE Symposium on, pages 324–329, October 1998.

54. Xu, X., Ansari, R., and Khokhar, A.: Power-efficient algorithms for Fourier analysis over random wireless sensor network. pages 109–115. 8th IEEE DCOSS, May 16-18 2012.

55. Kunis, S., Keiner, J., and Potts, D.: The nonequspaced FFT. www-user.tu-chemnitz.de/ skunis/paper/lecturenfft.pdf.

56. Potts, D., Steidl, G., and Tasche, M.: Fast fourier transforms for nonequispaced data: A tutorial. Modern Sampling Theory Mathematics and Applications, pages 1–23, 2001.

CITED LITERATURE (Continued)

57. Kunis, S.: Nonequispaced FFT - Generalisation and Inversion. Shaker Verlag GmbH, 2007.

58. Xu, X., Ansari, R., and Khokhar, A.: Power-efficient Hierarchical Data Aggregation using Compressive Sensing in WSN. IEEE ICC, June 2013. Preprint.

59. Xu, X., Ansari, R., and Khorkhar, A.: Power-efficient Hierarchical Data Aggregation using Compressive Sensing in WSNs. In IEEE International Conference on Communications (ICC), Budapest, Hungary, June 9-13 2013.

60. Artiola, J., Pepper, I., and Brusseau, M.: Environmental Monitoring and Characterization. Academic Press, 2004.

61. Clarke, R.: Relation between the karhunen loeve and cosine transforms. Communications, Radar and Signal Processing, IEE Proceedings F, 128(6):359 – 360, Nov 1981.

62. Baron, D., Wakin, M. B., Duarte, M. F., Sarvotham, S., and Baraniuk, R. G.: Distributed Compressed Sensing. Technical report, Electrical and Computer Engineering Department, Rice University, 2006.

63. Zhang, Y., Roughan, M., Willinger, W., and Qiu, L.: Spatio-Temporal Compressive Sensing and Internet Traffic Matrices (Extended Version). In SIGCOMM, pages 267–278, New York, NY, USA, 2009. ACM.

64. Xu, X., Ansari, R., and Khorkhar, A.: Adaptive Hierarchical Data Aggregation using Compressive Sensing (A-HDACS) for Non-smooth Data Field. In IEEE International Conference on Communications (ICC), pages 65–70, June 2014.

65. Candès, E. J. and Recht, B.: Exact matrix completion via convex optimization. Foundations of Computational Mathematics, 9(6), 2009.

66. Vuran, M. C., Akan, O. B., and Akyildiz, I. F.: Spatio-temporal correlation: Theory and applications for wireless sensor networks. Comput. Netw., 45(3):245–259, June 2004.

67. Compressive sensing. http://en.wikipedia.org/wiki/Compressed_sensing.

68. W.R., H., A., C., and H., B.: Energy-efficient communication protocol for wireless microsensor networks. Jan 2000.

CITED LITERATURE (Continued)

69. IEEE 802.15.4. http://www.ieee802.org/15/pub/TG4.html, 2014.

70. Cheng, J., Ye, Q., Jiang, H., Wang, D., and Wang, C.: STCDG: An Efficient Data Gathering Algorithm Based on Matrix Completion for Wireless Sensor Networks. IEEE Transactions on Wireless Communications, 12(2), 2013.

71. The Reduction Formula. www.math.niu.edu/ richard/Math230/reduction.pdf.

72. Candès, E. and Romberg, J.: Sparsity and incoherence in compressive sampling. Inverse probability, 23(3):969–985, 2007.

73. National Data Buoy Center. http://tao.noaa.gov/tao/data_download/search_map.shtml, 2014.

74. Figueiredo, M. A. T., Nowak, R. D., and Wright., S. J.: Gradient projection for sparse reconstruction:application to compressed sensing and other inverse problems application to compressed sensing and other inverse problems. IEEE J. Selected Topics in Signal Processing:Special Issue on Convex Optimization Methods for Signal Processing, 1(4):586–598, 2007.

75. Chiasserini, C. F. and Rao, R. R.: On the concept of distributed digital signal processing in wireless sensor networks. page 260. IEEE MILCOM, October, 2002.

76. Culler, D., Estrin, D., and Srivastava, M.: Overview of Sensor Networks. IEEE Computer Society, August 2004.

77. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E.: A Survey on Sensor Networks. IEEE Communication Magazine, pages 102–114, August 2002.

78. Kwon, S.: Analysis of Shortest Path Routing for Large Multi-Hop Wireless Networks. IEEE/ACM transaction of networking, 17(3), June 2009.

79. Xu, X., Ansari, R., and Khokhar, A.: Power-efficient hierarchical data aggregation using compressive sensing in wsns. In Communications (ICC), 2013 IEEE International Conference on, pages 1769–1773, June 2013.

80. Xu, X., Ansari, R., and Khokhar, A.: Power-efficient algorithms for Fourier analysis over random wireless sensor network. pages 109–115. 8th IEEE DCOSS, May 16-18 2012.

CITED LITERATURE (Continued)

81. Xu, X., Ansari, R., and Khokhar, A.: Power-efficient Hierarchical Data Aggregation using Compressive Sensing in WSN. In Wireless Communications and Networking Conference (WCNC), 2013 IEEE, pages 4381–4386, Oct 2013. Preprint.

82. Gilbert, A., Strauss, M., Tropp, J., and Vershynin, R.: One sketch for all: Fast algorithms for compressed sensing. San Diego, June 2007. 39th ACM Symp. Theory of Computing.

83. Cheng, J., Ye, Q., Jiang, H., Wang, D., and Wang, C.: STCDG: An Efficient Data Gathering Algorithm Based on Matrix Completion for Wireless Sensor Networks. IEEE Transactions on Wireless Communications, pages 850–861, 2013.

84. Kong, L., Xia, M., Liu, X.-Y., Wu, M.-Y., and Liu, X.: Data loss and reconstruction in sensor networks. In INFOCOM, 2013 Proceedings IEEE, April 2013.

85. Goldenberg, D. K., Lin, J., Morse, A. S., Rosen, B. E., and Yang, Y. R.: Towards mobility as a network control primitive. Roppongi, Japan, May 24-26 2004. Mobihoc.

86. Haar wavelet. http://en.wikipedia.org/wiki/Haar_wavelet.

87. Xu, X., Ansari, R., Khorkhar, A., and Vasilakos, A.: Hierarchical Data Aggregation using Compressive Sensing (HDACS) in WSNs. ACM Transactions on Sensor Networks, 11(3), August 2015.

88. Merry, R.: Wavelet Theory and Applications: A literature study. Technical report, Technical report, Eindhoven University of Technology, 2005.

89. Gilbert, A., M. Strauss, J. T., and Vershynin, R.: Algorithmic linear dimension reduction in the l1 norm for sparse vectors. 44th Annu. Allerton Conf. Communication, Control, Computing, August 2006.

90. Daubechies, I., Defrise, M., and Mol, C. D.: An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Comm. Pure Appl. Math., 57:1413–1457, 2004.

91. Tang, X. and Xu, J.: Optimizing Lifetime for Continuous Data Aggregation With Precision Guarantees in Wireless Sensor Networks. IEEE Transactions on networking, 16(4):904–917, August 2008.

CITED LITERATURE (Continued)

92. Zhang, H. and Shen, H.: Balancing Energy Consumption to Maximize Network Lifetime in Data-Gathering Sensor Networks. IEEE Transactions on Parallel and Distributed Systems, 20(10):1526–1539, October 2009.

93. Jiang, H., Jin, S., and Wang, C.: Prediction or Not? An Energy-Efficient Framework for Clustering-Based Data Collection in Wireless Sensor Networks. IEEE Transactions on Parallel and Distributed Systems, 22(6):1064–1071, June 2011.

94. Canli, T., Cupta, A., and Khokhar, A.: Power Efficient Algorithm for Computing Fast Fourier Transform over Wireless Sensor Networks. pages 549–556. IEEE Computer Systems and Applications, March 8, 2006.

95. Bientinesi, P., Pitsianis, N., and Sun., X.: Parallel 2D FFTs on the Cell Broadband Engine. April 2007.

96. Rajagopalan, R. and Varshney, P. K.: Data aggregation techniques in sensor networks: A survey. IEEE Communications Surveys and Tutorials, 8(4), 2006.

97. Dutt, A.: Fast Fourier Transform for Nonequispaced Data. SIAM Journal on Scientific Computing, 14(6):1368–1393, 1993.

98. Nealen, A.: An As-Short-As-Possible Introduction to the Least Squares, Weighted Least Squares and Moving Least Squares Methods for Scattered Data Approximation and Interpolation. Computer Methods in Applied Mechanics and Engineering, 2003.

99. Potts, D., Steidl, G., and Tasche, M.: Fast Fourier transforms for nonequispaced data: A tutorial. Modern Sampling Theory Mathematics and Applications, pages 1–23, 2001.

100. P.Thévenaz, T. B. and Unser, M.: Image Interpolation and Resampling, pages 393–420. Academic Press, 2000.

101. Kunis, S., Keiner, J., and Potts, D.: The nonequspaced FFT. www-user.tu-chemnitz.de/ skunis/paper/lecture_nfft.pdf.

102. Kunis, S.: Nonequispaced FFT - Generalisation and Inversion. Shaker Verlag GmbH, Germany, Feb 9 2007.

VITA

NAME: Xi Xu

EDUCATION: B.A.,Electrical and Computer Engineering, Jilin University, Changchun, China, 2005

Ph.D., Electrical and Computer Engineering, University of Illinois at Chicago (UIC), Chicago, Illinois, 2005

EXPERIENCE: Research Assistant, Multimedia Systems Lab, ECE Department, UIC, 2010

Image Research Intern, Advanced Technology Group Division at Dolby Laboratories, Inc., Sunnyvale, California, 2013

PUBLICATIONS: X. Xu, R. Ansari, A. Khokhar, "Power-efficient Spatio-Temporal Hierarchical Data Aggregation using Compressive Sensing in WSNs ", on submitting, ACM Transactions on Sensor Networks.

X. Xu, R. Ansari, A. Khokhar, "Parallel Nonuniform Discrete Fourier Transform (P-NDFT) Over Random Wireless Sensor Networks", on submitting, IEEE Transactions Parallel and Distributed Systems.

X. Xu, R. Ansari, A. Khokhar, and Athanasios Vasilakos "Hierarchical Data Aggregation using Compressive Sensing (HDACS) in WSNs", ACM Transactions on Sensor Networks 11, 3 August, 2015

X. Xu, R. Ansari, and A. Khokhar, Spatio-Temporal Hierarchical Data

Aggregation using Compressive Sensing (ST-HDACS). June 10-12, IEEE DCOSS, 2015

X. Xu, R. Ansari, and A. Khokhar, "Adaptive Hierarchical Data Aggregation using Compressive Sensing (A-HDACS) for non-smooth data field", IEEE ICC, 2014

X. Xu, R. Ansari, and A. Khokhar, Power-efficient algorithm for Fourier analysis over random wireless sensor networks. May 16-18, IEEE DCOSS, 2012, pp 109-115

X. Xu, R. Ansari, and A. Khokhar, Power-efficient Nonuniform 2-D Fourier Analysis using Compressive Sensing in WSN, April 7-10, IEEE WCNC, 2013, pp 4381-4386.

X. Xu, R. Ansari, and A. Khokhar, Power-efficient Hierarchical Data Aggregation using Compressive Sensing in WSN, June 10-12, IEEE ICC, 2013.