Energy Efficient In-Network Data Indexing and Query Processing for Wireless Sensor Networks

BY

MOHAMED MOHAMED ALI MOHAMED
B.Sc., Cairo University, 2007
M.Sc., University of Illinois at Chicago, 2009

THESIS

Submitted as 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:
Ashfaq Khokhar, Advisor
Milos Zefran, Chair
Rashid Ansari
Ajay Kshemkalyani, Computer Science
Goce Trajcevski, Northwestern University

To my Beloved and my parents, whose love nourishes my soul & heart.

ACKNOWLEDGMENTS

Throughout the past years, I have been surrounded with numerous supportive individuals. I am deeply grateful for Prof. Ashfaq Khokhar, my advisor and dear mentor who guided me wisely and gently through the research maze. Prof. Khokhar helped me directly and indirectly not only to get over the obstacles I faced, but also taught me how to independently handle them. His openness to various ideas, challenging and debating them, taught me how the evaluation process of ideas has to be a thorough one that does not make judgments before full analysis. Beside his technical mentorship, I enjoyed sharing some common interests with him as playing squash, reading poetry, and enjoying nice conversations.

The two supportive gentlemen whom I was honored to know as my longest roommates and labmates were Dr. Fahd Saeed and (soon to be) Dr. Omar Cheema. Fahd was one of the most fun people I knew. The kind of person that disagreeing to anything he says is a pleasure, as it leads to mentally challenging conversations that used to last quite long while having some nice chai. Omar is of the most socially skillful people, who is able to be extremely polite while doing everything he wants in a pretty quiet and laid back manner. Saeb Ahsan was my first roommate from whom I learned so many things, being new to Chicago. Beside the fact that Saeb was always able to create a stream of challenging unexpected questions at any moment.

My first labmates Dr. Mohamed Hefeida, Dr. Fadi Almasalha, Dr. Turkmen Canli formed my amazing working environment. One of which, where if Mohamed was present, the working environment can be smoothly transformed to a fun place that might motivate anything but

work. Fadi who was my best gym buddy, and a very effective person in whatever he works on. Turkmen was the most reasonable labmate. The one that one would learn a lot about dedication just by observing him. Also I would like to thank my latter labmates for the wonderful time I spent with them; (All Dr. to be) Yasaman, Xi, Kamran, Nazaneen, and Ansab.

Several faculty members were very supportive and helpful for me. Prof. Goce Trajcevski from Northwestern University was a man whom I learned so much from. He has been very supportive and taught me a lot about productivity. He is also a man whose company made me enjoy good time on several conference travels. Prof. Rashid Ansari was a great resource in the technical field for me. His humble personality and work ethics were a great resource for me to learn, without him even giving explicit advices. I am also grateful to my defense committee, Prof. Milos Zefran and Prof. Ajay Khsemkalyani who taught me multiple courses throughout my Ph.D. I had the honor and pleasure to work as a teaching assistant for long time with Mr. Robert Becker, one of the most exceptional teachers I ever met, and Prof. Vladimir Goncharoff who is a great professor and an amazing human being.

The ECE department staff was a great asset that helped me be focused on my work, by them taking care of a lot of details in the background. The amazing Erica, Beverly, Ala and Tina who were always supportive in everything I needed from the first to the last day of my stay. Also the dear Rejona who was the smiley face of every beginning of a working day. The UIC Office of International Students was another great asset that always guided me through its supportive staff and amazing programs.

In UIC I would like to highly acknowledge the place at which most of the research ideas came

to me. It was not a usual place to do research, as a lab, office or library. The swimming pool in UIC was my place of research inspiration, where in a few laps along with thinking of a research problem, I had to rush out in order to start sketching a new idea on paper and go through its evaluation.

Over the course of my academic stay, the city of Chicago has blessed me with a treasure of unique friends, teachers and mentors whose value for me cannot be expressed in words, but, their special place is in my heart. These years were great time in a lovely place with amazing people.

MMA

ACKNOWLEDGMENTS (Continued)

Contribution of Authors

This work has been under the supervision of my advisor Prof. Ashfaq Khokhar, and in collaboration with Prof. Goce Trajcevski who were greatly contributing into all the details of the ideas, algorithms and techniques. Prof. Rashid Ansari and Prof. Aris Ouksel collaborated in the development of the main data indexing algorithm for static cases, and provided a lot of feedback that helped greatly in enhancing the algorithm, and giving it more depth and robustness.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

FIGURE **PAGE**

LIST OF FIGURES (Continued)

113

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

BSP	Binary Space Partitioning
CO	Carbon Monoxide
CVT	Centroidal Voronoi Tessellation
GHT	Geographic Hash Tables
GPS	Global Positioning System
JiST	Java in Simulation Time
MAC	Media Access Control
MRF	Mobility Resolution Factor
NRMSE	Normalized Root Mean Square Error
OBT	Orthogonal Bisection Trees
QoS	Quality of Service
SBDR	Structure-Based Distributed Request
SPDR	Structure and Proximity based Distributed Request
SWANS	Scalable Wireless Ad hoc Network Simulator
UIC	University of Illinois at Chicago
WSN	Wireless Sensor Networks

SUMMARY

Advances in miniaturization of devices equipped with sensing, computing and communication capabilities have spurred significant interest in Wireless Sensor Networks (WSNs) as a tool for distributed data gathering, field estimation, and query processing. WSNs provide the capability of monitoring any given physical phenomena, reporting up to date information to interested users, and reacting to the observed phenomenon using predetermined trigger mechanisms. Energy efficiency has been one of the main concerns in the design and use of WSN based applications, as replenishing power to sensor nodes is impractical or not possible, particularly when they are deployed in harsh environments hostile terrains, or human-unfriendly locations. The focus of this dissertation is the problem of organizing sensed information (also referred to as data indexing) for efficient in-network query processing in context of static as well as mobile networks.

Existing solutions for the data indexing problem can be classified from several dimensions. Some of these solutions rely on a centralized approach, where data across the network is transmitted to one sink node, and it is organized at that node. Subsequently, all the queries are processed at this sink node. Such an approach suffers from its poor scalability to large networks and inherent highly non-linear increase in its traffic towards sink node. Decentralized solutions rely on in-network organization of the data. Data organization in these solutions is either optimized for processing of queries related to: 1) the sensed values; or 2) the locations of the sensed values. Therefore application that entertain both types of queries face disparity in efficiency.

As for the maintenance of the data-indexing system is concerned, some of the existing solutions rely on transmitting data in raw form across the network, while other solutions construct data models, in order to decrease the amount of transmitted information. The former method gives WSNs the capability of answering queries with same accuracy across the indexing hierarchy, at the cost of a significant increase in the data traffic. The latter category of solutions, however, supports approximate response to queries, providing the benefit of low maintenance cost for the system.

In this dissertation, we present an energy and time efficient distributed data indexing system, which supports approximate querying, relying on novel data models that represent the sensed values and sensor nodes locations independently. The presented system is capable of answering different query types with equal efficiency. It also requires minimal maintenance cost, as it employs fixed size update messages across the indexing hierarchy. Our solution does not create traffic congestion around the sink node, and hence, prolongs network life time. The presented abstraction techniques are data structure independent, which gives the flexibility of building the indexing system on a set of widely used binary space partitioning data structures. Our experiments show the efficiency of the presented methods to capture different types of phenomena and answer queries in a better way, compared to the existing state of the art.

The presented distributed data indexing system is capable of handling mobility of sensor nodes within the sensed field, without incurring any significant overhead on the energy cost, or query processing performance. It is capable of handling the majority of sensor nodes mobility within their own localities. We also present a novel energy and time efficient methodology for man-

aging the sensor nodes mobility, in order to redistribute resources within the sensed field in response to occurrence of specific events.

CHAPTER 1

INTRODUCTION

Wireless sensor networks (WSN) emerged as a useful tool in several realms (1). They can be represented as a distributed system of sensor nodes. A sensor node is capable of sensing a physical phenomenon, while also having extended capabilities for storage, computation, and wireless communication. WSN are beneficial to use in various applications, especially ones for which human access is limited, as in environmental, industrial, and military applications. A WSN gets deployed in the desired location, which we call the sensed field. The sensor nodes start discovering each other and connecting according to a defined procedure. They then start sensing the field and reporting their data to a defined –one or more– node(s). The network user can dispatch queries to the network inquiring information about the sensed phenomena. In some scenarios, the sensor nodes are capable of moving after their initial deployment, in order to track a target, or to have better coverage for the sensed field.

The main task of a WSN is to monitor the sensed field, and provide up-to-date information about it. An in-network storage shall be designated for this information in a way that facilitates answering queries received by the network. In order for the WSN to perform its task efficiently, sensed data needs to be organized in a way that enables: 1. maintaining updated information across the network; and 2. responding to received queries in a quick manner. Energy consumption is a big challenge in the WSN field (2). The radio communication of sensor nodes is the most energy consuming process. Hence, a desired data organization scheme for WSN is required

to be energy efficient, and in the same time able to achieve time efficient query processing. Due to the the usual deployment of the sensor motes in fields having physical imperfections, and due to the natural error in the sensor motes, the reported values by the sensor nodes are not considered exact values. They rather represent the sensed phenomenon within some considered precision error.

Research in energy-aware query processing has been evolving over the past decade. Plethora of algorithms have been presented trying to build a system that is capable of retrieving sensed information, creating an aggregated in-network –point or distributed– storage for it, and hence, responding to queries accordingly. With a wide variation of ideas trying to solve this problem, data indexing becomes in focus for all of them. Data indexing in WSN aims at creating an in-network communication and storage methodology, which assists the network fulfilling its responsibilities. As previously stated, the main challenge that any data organization/indexing scheme faces in WSN is to perform its job with the least possible energy consumption. The most participating factor in energy consumption is the wireless communication (2). Thus, this two extremes problem can be phrased as: *sharing as much useful information with least possible communication.*

Existing data indexing algorithms can be classified from different dimensions. Some of them follow a centralized aggregation method, where all information are pulled towards the sink node; which is the central node connected to the base station. The methods applied by this first set usually suffer from load balancing issues. The other set creates an in-network logical data structure from the sensor nodes. In the latter set, each sensor node is logically connected

to one –or more– indexing node(s) of that data structure, and accordingly reports its sensed information to the indexing node(s). Some data indexing algorithms perform further aggregation for the gathered information, aiming at creating hierarchical representation for the field. The reported information can be categorized as the sensed value of the phenomenon, and the respective location at which the values were sensed. Each of these categories has its significance depending on the application.

The existing work in data indexing is yet incapable of creating a system that aggregates the different sensed information types with equal efficiency, while keeping low energy cost for maintaining the data structure up-to-date. Also, the aggregation techniques applied are either trading-off the two information categories (sensed values and their respective locations), or designed to better suit a specific data distribution of the sensed phenomenon. Most of the existing data indexing algorithms do not put mobility of sensor nodes into consideration. However, in many WSN applications, the sensor nodes are given the freedom of changing their locations in the field to better achieve adaptability to dynamic behavior of the sensed phenomena, or to track objects in the sensed field. In these scenarios, if the indexing algorithm were not to adapt to such mobility, the network might incur significant increase in energy consumption; Or worse, have the sensor nodes disconnected from the indexing node(s) they are –logically– connected to.

1.1 Problem Statement

We define the data indexing and querying problem as the requirement of an in-network data indexing and query processing system that is capable of achieving an up-to-date storage for the

network information with low maintenance energy cost, and responding to received queries with low latency and processing energy cost. The required solution shall be able to efficiently handle different types of information in the network in a generic way that does not presume configurations in different dimensions. It also has to prolong the network lifetime by lowering the maintenance cost of the indexing structure and keeping a balanced workload between the indexing nodes.

1.2 Contribution of this Dissertation

In this dissertation we design and implement a storage and communication efficient solution for the in-network data indexing and query processing problem in wireless sensor network. The solution provides a generic energy efficient methodology to handle network information. It consists of:

- **Data Abstractions:** Novel abstraction techniques for both sensed values and sensor nodes locations. The abstraction scheme equally handles both types of information, and aggregates them in an energy efficient manner, providing a hierarchical in-network storage that is capable of answering different queries with low latency, and further able to provide immediate answers to approximate queries and some types of exact queries. The presented abstraction techniques are independent from the underlying phenomenon distribution, and hence better apply to various data distributions keeping load balancing into consideration, as we will show. A version of this work can be found in (3; 4). The abstraction techniques are presented in detail in Sections 3.1 and 3.2.

- **Indexing Structure:** In order to prove that the abstraction techniques are generic enough to fit a wide variety of the commonly used spatial data structures, we present the applicability of the algorithm on two data structures:

    1. A K-D Tree (5; 6) implementation, which manifests the use of any binary space partitioning (BSP) based data structures. A version of this work can be found in (4). Details are discussed in Section 3.3.
    2. A Voronoi Treemap (7; 8), which is a hierarchical form of Voronoi diagrams (9), a highly efficient spatial partitioning for WSN. A version of this work can be found in (10). Details are discussed in Section 3.4.

- **Query Processing:** We present an energy efficient, logarithmic in time, multi-attribute query processing algorithm that is capable of analyzing the query types and constraints, and forwarding each query only to the appropriate indexing nodes. The query processing algorithm supports approximate querying, where it can respond to queries within a given error bound. It can provide immediate responses for the extreme (*max, min*) sensed values. The algorithm handles queries about the sensed values and queries about the sensor nodes locations with equal efficiency. A version of this work can be found in (4; 10). The query processing is presented in Section 3.5.

- **Mobility Management:** We show how the presented algorithms can efficiently adapt to different spatial configurations of the sensor nodes under mobility, without incurring extra overhead out of the areas experiencing mobility. The mobility algorithm applies

to all BSP data structures. A version of this work can be found in (11). The mobility management is discussed in Chapter 4.

- **Resource Distribution:** We present efficient methodologies for scalable management of relocation of mobile sensors in WSNs, in response to a detection of event of interest. The presented work takes into consideration the minimum nodes count needed in each spatial region for guaranteeing certain QoS criteria. Capitalizing on a hierarchical structure, we present distributed protocols which improve both the response time and the energy consumption due to communication, along with the choices of nodes to move seeking the optimization of the traveled distance. The presented approaches are capable of handling simultaneous detection of multiple events. A version of this work can be found in (12). Resource distribution is presented in Chapter 5.

- **Implementation and Performance Evaluation** The presented methods were implemented on top of the SIDnet-SWANS WSN simulator (13) based on Jist-SWANS discrete event simulation engine (14). The data abstractions are implemented for K-D trees and Voronoi Treemaps. Robust performance analysis is performed for the effect of each data structure in the data indexing. Mobility is simulated with several mobility models, and their performance is evaluated. Our experimental results show the efficiency of the presented algorithm, in terms of query latency, and maintenance cost. Experimental results are presented in Chapter 6.

1.3 Document Organization

This dissertation is organized as follows: In Chapter 2, we present a discussion about the preliminary assumptions and the analysis of the different dimensions of the problem. After this we present the data abstraction techniques and their application to different data structures, and the query processing methodology, in Chapter 3. In Chapter 4 we present the mobility handling algorithm. We present a set of algorithms for resource distribution in Chapter 5, followed by the experimental results presented in Chapter 6. We discuss the related work in Chapter 7. Chapter 8 concludes the dissertation and Chapter 9 outlines the directions of the possible future work.

CHAPTER 2

PRELIMINARIES

In this chapter we describe the general setting of the presented system. We show the required features in any data structure to which the presented abstraction methods can be applied. We then present the notations that will be used to describe the queries to the indexing system. After that, we present a discussion about the different dimensions of the data indexing problem, in order to quest the metrics to be used for the assessment of the presented system. We follow this with a discussion of the related work.

2.1 Data Structure Assumptions

The presented abstraction needs to work on top of a hierarchical spanning tree. The spanning tree is to be rooted at the sink node. The tree has to conform to any spatial data structure that splits the given space in multiple granulation levels by creating contiguous non-overlapping regions, and without producing holes. A widely used group that holds these features is the Binary Space Partitioning (BSP) data structures, which recursively subdivide the space into convex sets using hyperplanes, e.g, KD-trees, Quad-trees, Octrees (5; 6). We presume the existence of the indexing tree in a balanced form before starting the data indexing system. Figure 1 depicts an color-coded example of an orthogonal bisection based KD tree.

In the used indexing data structure, each leaf node know the borders of the spatial region it covers. Each leaf node is considered responsible of the sensor nodes within its region. We will be

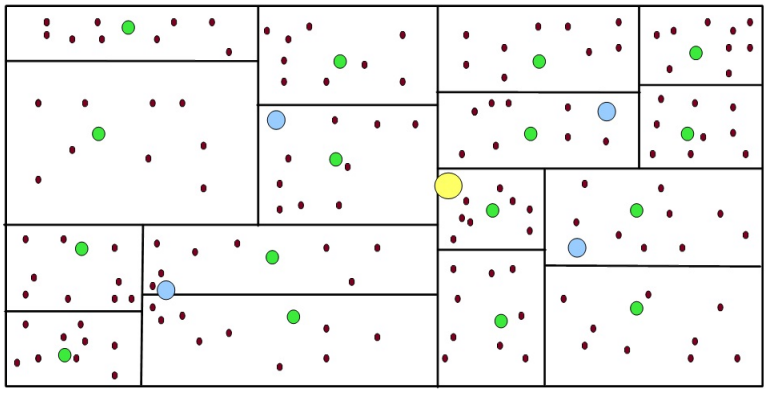


Figure 1. An OBT 160 node WSN with 16 local cluster heads (Green), 4 next level cluster heads (Blue), and one yellow sink node.

using the name local cluster head interchangeably with the indexing leaf node throughout this document. Each sensor node will be logically connected to a local cluster head (i.e, indexing leaf node). The local cluster heads shall be responsible for gathering information from their cluster nodes, and applying the first phase of the data abstraction method.

2.2 Query Types

One main task of a WSN is to respond to queries of its administrator. The queries may inquire values of the sensed phenomena, either in the whole field or in a specific region. They may also inquire the location from which a value, or a range of values, were reported. From an information perspective, the network administrator is likely to be interested in more precise information about special events, as the extreme values (maximum or minimum) and their locations. However, for the rest of network information, queries are more likely to inquire information about the overall behavior rather than specifics. Also, the reported values of sensor

nodes are generally not accurate due to imperfections and other physical aspects. Because of this, approximate queries are well suiting for WSN, where the query contains a field to specify the accuracy level accepted for the answer. This applies more to the overall queries than the extreme values queries. To capture these properties, queries are considered as predicates with attributes, as follows:

$\boldsymbol{Q(P,T,\ C,\ A)}$, where:

- $\boldsymbol{P}$ denotes the sensed phenomenon (e.g., Temperature, Humidity)
- $\boldsymbol{T}$ denotes a type. We will denote the sensed values with $\boldsymbol{T = v}$ and locations type with $\boldsymbol{T = l}$.
- $\boldsymbol{C}$ denotes the type of search bounds for the query: geometric bounds within the sensed field $\boldsymbol{(G)}$, and/or, either value range within the sensed values $\boldsymbol{(R)}$ or an extreme ($\boldsymbol{M}$, where $\boldsymbol{M = min}$ or $\boldsymbol{M = max}$).
- $\boldsymbol{A}$ denotes the required level of accuracy for the query response.

An example of a physical-space query with range constraint would be:

Q(Temperature, v, [70 °C**, 80 °C*****], {*****[0, 0],[30, 50]}, 80%)***

Which can be straightforwardly translated to an SQL-like syntax:

SELECT TEMPRATURE_VALUES	T=v
BETWEEN 70 °C TO 80 °C	R=[70 °C,80 °C]
INSIDE RECTANGLE [0, 0],[30, 50]	G=[0, 0],[30, 50]
WITH ACCURACY = 80%	A=80%

An example of a data-space query with range constraint is:

Q(Temperature, l, [70 °C**, 80 °C*****], {*****[5, 7],[30, 10]}, 65%)***

2.3 Data Indexing Analysis

In this subsection we present an analysis of some metrics that an indexing system has to conform to, in order to be efficient. We shall use these metrics throughout this document to evaluate previous contributions in solving the indexing problem as well as the presented solution.

Metric 1: WSN Information Representation

WSN information can be classified as: 1. sensed values; and 2. sensor nodes locations. Each of these categories represent a different domain for which the network can be viewed. The sensed values represent the readings of the sensed phenomenon across the whole spatial area of the sensed field, or part(s) of it. The sensor nodes locations represent the locations of sensors that are reading values of the whole possible range of values of the sensed phenomenon, or part(s) of it. Different WSN applications have interest in both types of information. In order for a

data indexing system to be generic enough to satisfy the various needs of WSN applications, it should not have any correlation assumptions between the two categories. In other words, the sensed values and sensor nodes locations have to be considered orthogonal, and hence treated independently across the indexing system, in a way that enables it to respond to any type of query that involves any permutation of constraints on both categories. Also, a data indexing system should not have prior assumptions for the distribution of either the sensor nodes locations in the spatial domain, or the sensed values in the data domain. It rather should be able to adapt to any distribution in a way that enables it to function with the same efficiency.

Metric 2: Load Balancing

In order to prolong the network lifetime, the indexing system should equally distribute the workload over the indexing structure. This balance should be applied horizontally and vertically in a hierarchical indexing structure. Horizontal load balancing means that at each level of an indexing the amount of information shall be equally distributed among the indexing nodes, for the spatial regions or data ranges they cover. Vertical load balancing refers to the consistency in the amount of information transferred between the levels of the indexing tree. However upper level nodes might be considered to cover a larger amount of indexing nodes, and hence are expected to receive more information, the increase in information in this fashion creates traffic bottlenecks towards the upper level nodes. These traffic bottlenecks along with the large data transfer consumes the energy for the upper level nodes, and decrease the network lifetime. A solution for the vertical load balancing problem that benefit from approximate querying and creates a multi-resolution indexing by using modeling was first presented in (15). Data

abstraction either through a specific model or using any similar abstraction technique enables overcoming the vertical unbalance in the indexing tree. It also creates a hierarchical representation of the field, where the upper level indexing nodes store granular information about larger parts of the field, while the lower level indexing nodes store more detailed information about smaller parts of the field.

Metric 3: Maintenance Cost

An efficient indexing system should have an energy efficient maintenance strategy to update the network information. It should preserve the locality of updates, where a sensor node is not required to report its information to an indexing node that is spatially distant from its location. This is because the multi-hop communication is costly in terms of scheduling and actual data transmission. The information at each sensor node, which we call raw data, should not be redundantly transmitted in the indexing structure. But rather, it should be reported once to an indexing node, then abstraction shall take place to represent the raw data of multiple sensors at different levels of the indexing structures.

Metric 4: Query Processing Time

The query processing time represents the time between the dispatching of a query to the network, till the response is received. This includes the processing of the query, regardless of the arrangement of information inside the network, or the presence/absence of an indexing structure.

The solution for the data indexing problem lies between two extremes: a centralized solution, or a fully distributed solution. In a centralized solution the maintenance cost of the network is

quite expensive, as all information has to be gathered to one central node. This also increases the traffic towards this central node, which accordingly decreases the network life time due to this unbalance. In a fully distributed solution, each sensor node is considered an indexing node for its own information. However this eliminates the cost of updating information across the system, it requires for any query to be answered that the query gets flooded across the whole network. In such way a significant cost is incurred to respond to each query. Looking at this dimension, a good solution for the problem is the one that minimizes the maintenance (i.e, update) cost, and becomes able to direct a query to the specific node(s) capable of providing a satisfying answer for it.

CHAPTER 3

DATA INDEXING AND QUERY PROCESSING

Chapter Threee: Data Indexing and Query Processing (Previously published as Mohamed, M. M. A. and Khokhar, A. A.: Dynamic indexing system for spatio-temporal queries in wireless sensor networks. In Mobile Data Management (MDM), 2011 12th IEEE International Conference on, volume 2, pages 35-37. IEEE, 2011.; Ali Mohamed, M. M., Khokhar, A., Trajcevski, G., Ansari, R., and Ouksel, A.: Approximate hybrid query processing in wireless sensor networks. In Proceedings of the 20th International Conference on Advances in Geographic Information Systems, pages 542-545. ACM, 2012; Mohamed, M. M. A., Khokhar, A. A., and Trajcevski, G.: Voronoi trees for hierarchical in-network data and space abstractions in wireless sensor netowrks. In Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems, pages 207-210. 2013.)

We now present the details of the abstractions and their use for efficient query processing along with the nodes' behavior for receiving requests and, in response, processing the given queries. To efficiently respond to different query types a given WSN needs both physical-space and data-space abstractions, defined respectively as follows:-

Physical-space Abstraction: Representing the sensed data in the field of interest at multiple scales with respect to the geographical location of the sensing nodes.

Data-space Abstraction: Representing the sensor nodes' locations in the field of interest at multiple scales with respect to the range of the sensed data values.

These abstractions must be performed in a manner that enables seamless aggregation as well as proper preservation of the heterogeneous the data types. Towards that, our main desideratum is to minimize the size of the updating messages, thereby reducing the communication costs and prolonging the overall networks lifetime (2). Clearly, decreasing the size of the messages while retaining the utility of the information content should be done in an energy-efficient manner from the perspective of the local computations too. Hence, the data flowing across a particular in-network hierarchical structure would have two forms: Raw data: The location and sensed value of each individual sensor node. Progressively refined: The approximate embodiment of the raw data for a geographic region or a data-space subset.

In the next subsections, we discuss the methods of processing and abstracting raw data and present our novel representation constructs for progressive refinement.

3.1 Physical-space Abstraction

The objective is to provide a hierarchical multi-resolution abstraction scheme, enabling the underlying hierarchical structure to answer approximate queries aimed at any fixed region of the sensed field. To achieve this, the sensed data is gathered locally by a representative node within each region, which then creates an abstract (coarse) representation of the sensed values. The abstract representations from multiple regions are then merged in a hierarchical fashion to represent larger regions in coarser forms. We formulate the physical-space abstraction problem as follows:

Given: A hierarchical spanning tree $\mathsf{T}(\mathsf{V}', \mathsf{E}')$ of depth $\mathtt{d}$, and *(w.l.o.g.)* a fixed fan-out $\mathtt{f}$, in a graph $\mathsf{G}(\mathsf{V}, \mathsf{E})$, $\mathsf{E}' \subseteq \mathsf{E}$, $\mathsf{V}' \subseteq \mathsf{V}$, where $\forall \mathtt{i}$:

- Each leaf node $v'_{d,i}$ represents a cluster (a non-overlapping spatial region), and
- It is logically connected to nodes v_{ij}, where $j = 1, 2, .., D$, and $\forall v_{ij} \in V$, $D = (|V|/f^d)$, s.t. $\cup_{i,j}^{f^d,D} v_{ij} = V$, and nodes v_{ij} are geographically collocated.

Find: An abstract representation $p_{v'_{d,i}}$ of the sensed values $R = \{r_1, r_2, .., r_D\}$ in the region associated with node $v'_{d,i}$, such that: *average error*, *communication cost*, and *computation cost* are decreased. We "loosely" assume existence of a spanning tree as the only "needed structure" – our presented methods are independent from the selection of a particular hierarchical indexing structure.

Each node $v'_{d,i}$ in the spanning tree $T(V', E')$ gathers the sensed values $R = \{r_1, r_2, .., r_D\}$ from its set of logically connected sensor nodes v_{ij}, in its vicinity, and stores them in an array. Upon acquiring its population's readings, each node $v'_{d,i}$ rank-orders the sensed data and stores it in an array along with their corresponding sensor nodes' physical locations.

3.1.1 Physical-space Representation at the Leaf Nodes of the Indexing Structure

The abstract representation $p_{v'_{d,i}}$ is introduced as an array of a fixed-size (k) and its values are chosen by regularly sampling the sorted array of the population nodes with interval (D/k), including the first and last elements of the sorted array. Figure 2 shows an example of the sensed values in a group of sensor nodes in a small network. Leaf nodes of the spanning tree, acting as local cluster-heads connected to their 1-hop neighbors, gather all the sensed values and create the corresponding arrays.

The regularly sampled array $p_{v'_{d,i}}$ captures the main features of its sensed values within each

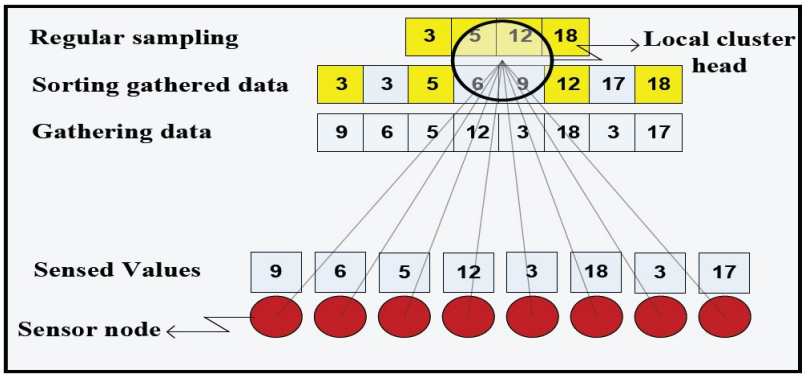


Figure 2. Sensor nodes (in red) transmit their readings to the leaf node (cluster-head), which sorts the received values and creates the representation construct by regular sampling.

cluster. It contains the lowest and highest readings for the phenomenon in its first and last elements, respectively. The distribution of readings and the capability of interpolating other values within acquainted error bound is determined by the size of the array. This approach mimics a curve fitting process and is generic enough to capture different phenomena.

3.1.2 Physical-space Representation at the Non-Leaf Nodes of the Indexing Structure

To develop abstract representations of larger regions represented by the non-leaf nodes of the spanning tree, each leaf node $v'_{d,i}$ can apply wavelet transformation (16) to its representation construct $p_{v'_{d,i}}$, and transmit the compressed representation to its parent in the spanning tree. Each non-leaf node receives from its children a set of f arrays, representing the physical-phenomenon at spatially non-overlapping regions in the field. Sample arrays are merged into a larger one representing the sensed phenomenon in the area enclosing the f regions. The new

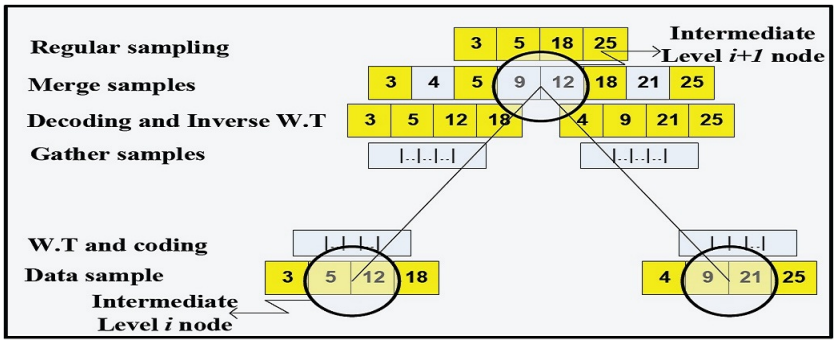


Figure 3. Physical-space merging process in an intermediate level of the spanning tree ($f = 2$). Lower level nodes compress their samples and transmit them to their parent which decodes, merges them and samples again with rate $1/f$.

physical-space abstraction of the larger region is created by regularly sampling the merged array with a given sampling interval (f). Upon completion, the new construct is of the same size as the received (input) arrays. It provides a regular sorted sample of the larger population, but in a coarser form. Figure 3 illustrates the physical-space across one intermediate level. The process of updating the data-payload throughout the participating nodes is performed by using fixed size messages, keeping the communication workload equally distributed.

When a physical-space query is received by any of the indexing structure nodes, it can respond within the level of accuracy it supports. The sorted data gives the capability of interpolating real sensed values using the sample array elements. Using inexpensive linear interpolation a node can determine the existence of a sensed data range, with a level of confidence relevant to its position in the hierarchy. A data elaborative example for physical-space abstraction across multiple levels of the indexing tree is shown in Figure 4.

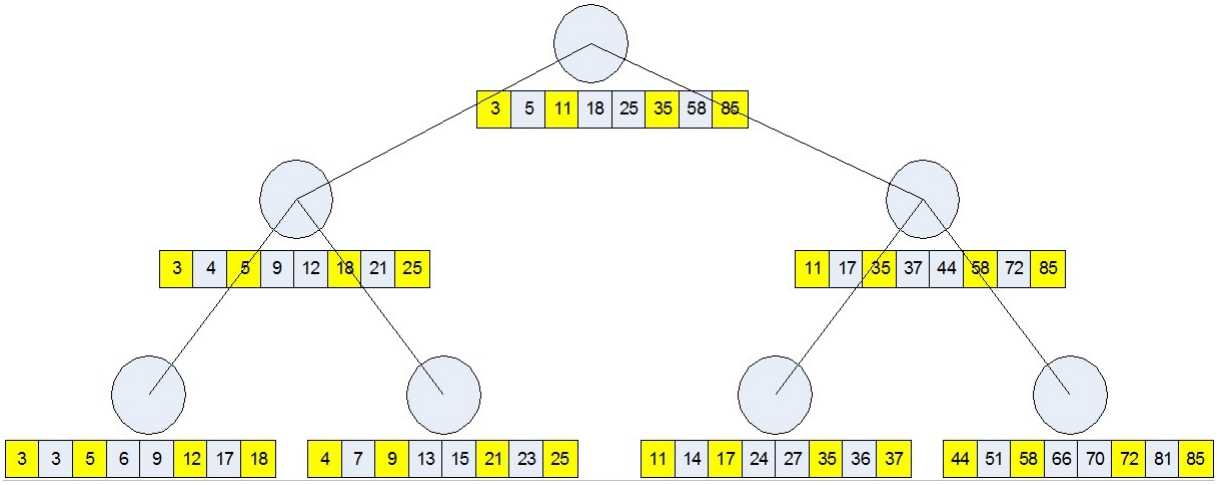


Figure 4. Physical-space abstraction across multiple levels of a hierarchy with $f = 2$. Each node keeps a sorted array of sensed values within its region and sends a sample of it (yellow background) to its parent, which merges all the samples from its children.

3.1.3 Error Bound Due to Hierarchical Sampling

A query response based on sampled values at any intermediate node in the indexing structure will be approximate – however, the error will be bounded. In our scheme it may be compounded due to the hierarchical nature of the sampling procedure. In the following we demonstrate that this error is within a factor of '*2*' compared to a centralized sampling scheme where all the sensed values of a given region are available.

In centralized settings, at any level j of the hierarchical structure, the distance between two samples is:

$$N^{j}_{Centralized} = \frac{f^{(d-j+1)}D}{k} \tag{3.1}$$

Accordingly, the maximum interpolation error for a query will be:

$$E^{j}_{Centralized} = \frac{f^{(d-j+1)}D}{2k} \tag{3.2}$$

In the presented hierarchical sampling the values can be skewed in position during the merging steps, in comparison to a centralized solution. This results in a shift affecting the representative sample values at each level of the hierarchy which, in turn, affects the representation accuracy. A similar analysis for a more special case of this idea was presented in (17). We formally define skewing limit as the maximum shifting of position of a sampled value during the hierarchical sampling process, compared to a centralized solution.

Lemma 1. *The error introduced due to the skewing limit at any level j of abstraction for intermediate sample values is no more than $\frac{f^{(d-j+1)}D}{k}$, (where k is the sample size).*

Proof. Using mathematical induction:

1. Base case ($j = d - 1$): f regular sample sets representing $f \times D$ population are merged and sampled for a new sample set S' of same size k. For each intermediate range between sampled elements i and $i-1$ in S', where $1 < i < k$, the number of elements less than $S'[i-1]$ is given by

$$lb = (i-2)\frac{fD}{k} \tag{3.3}$$

While the number of elements greater than $S'[i]$ is given by

$$ub = (k-i)\frac{fD}{k} \tag{3.4}$$

Accordingly, from equations (3)and (4), the maximum number of elements between i and $i-1$ is given by

$$N^{d-1}_{Distributed} = fD - lb - ub = fD - \frac{fD}{k}(k-2) = 2\frac{fD}{k} \tag{3.5}$$

Therefore, the maximum introduced error at this level is given by

$$E^{d-1}_{Distributed} = \frac{fD}{k} \tag{3.6}$$

2. Inductive step: For abstraction at a lower level of the indexing structure, If the relationship holds for $j = m < d$ then at $j = m-1$: For each intermediate range between elements i and $i-1$ in the new sample S', where $1 < i < k$, the number of elements less than $S'[i-1]$ is given by

$$lb = ((i-2)\frac{fk}{k})\frac{f^{d-m}D}{k} = (i-2)\frac{f^{d-m+1}D}{k} \tag{3.7}$$

While the number of elements greater than $S'[i]S$ is given by

$$ub = ((k-i)\frac{fk}{k})\frac{f^{d-m}D}{k} = (k-i)\frac{f^{d-m+1}D}{k} \tag{3.8}$$

Accordingly, from equations (7) and (8), the maximum number of elements between i and $i-1$ is given by

$$N^{m-1}_{Distributed} = f^{d-m+1}D - \frac{f^{d-m+1}D}{k}(k-2) = 2\frac{f^{d-m+1}D}{k} \tag{3.9}$$

Therefore, the maximum introduced error at this level is given by

$$E^{m-1}_{Distributed} = \frac{f^{d-m+1}D}{k} \tag{3.10}$$

3. Therefore, For any level of depth j in the indexing tree ($1 \leq j \leq d$), the maximum number of elements between any two intermediate samples i and $i-1$ is given by

$$N^{j}_{Distributed} = 2\frac{f^{d-j+1}D}{k} \tag{3.11}$$

And the error at such level is given by

$$E^{j}_{Distributed} = \frac{f^{d-j+1}D}{k} \tag{3.12}$$

In conclusion, the error bound due to skewing in a distributed solution cannot exceed the size of one sampling distance in an alternative centralized solution. The size of sample set and the branching factor of the indexing tree are affecting parameters to this bound.

3.2 Data-Space Abstraction

We assume that the possible values of the sensed phenomenon are delimited within a finite range $[min, max]$ for the potentially queried data-space. Our objective is to provide a hierarchical multi-resolution abstraction scheme to obtain approximate answers to queries that involve localizations of the related sensor nodes.

Hence, the data-space within each physical region is divided into q ranges and each one is assigned to a representative node in the region. Each such node creates an abstracted representation depicting the locations of all the sensor nodes within a particular data range – e.g., locations of all the nodes within the region that have sensed temperature above 110 degrees and below 150 degrees. This data-space abstraction is performed by the nodes at different levels of the spanning tree corresponding to the regions represented by the nodes. Using the same definition of the spanning tree $T(V', E')$ within the graph $G(V, E)$ from the previous subsection,

the data-space abstraction aims to:

Find: A representation construction $L_{v'_{d,i}}$ of the sensor nodes location distribution *w.r.t* its sensed value for each leaf node $v'_{d,i}$, such that each of the *average error*, *communication cost*, and *computation cost* are decreased.

Similar to the physical-space abstraction, the data-space abstraction starts by leaf nodes $v'_{d,i}$ collecting their population's sensed values – along with the corresponding location where a particular value was sensed. Each leaf node sorts the gathered information according to sensed values, and stores them in an array.

Each set G_l of f sibling leaf nodes represents a group of neighboring clusters within region l, where $l = 1, 2, .., f^{d-2}$. The data-space in each region l is split into q data ranges, where $q \geq f$. The responsibility of the data-space in each region l gets distributed among the f leaf nodes of the group G_l. Each cluster head node (i.e: leaf node $v'_{d,i}$) is assigned the duty of keeping the position of any sensor, within its region l, that reads a value within its data range(s) of responsibility. Assuming, for simplicity, that each leaf node is responsible only for one data range ($q = f$), the ranges of the data space for a group G_l, can be expressed as:

$$RG_l = \{RG_{l0}, RG_{l1}, ..., RG_{lf}\} \tag{3.13}$$

$$RG_l = \{[min, V_1], [V_1 + \epsilon, V_2], [V_2 + \epsilon, V_3], ..., [V_f + \epsilon, max]\} \tag{3.14}$$

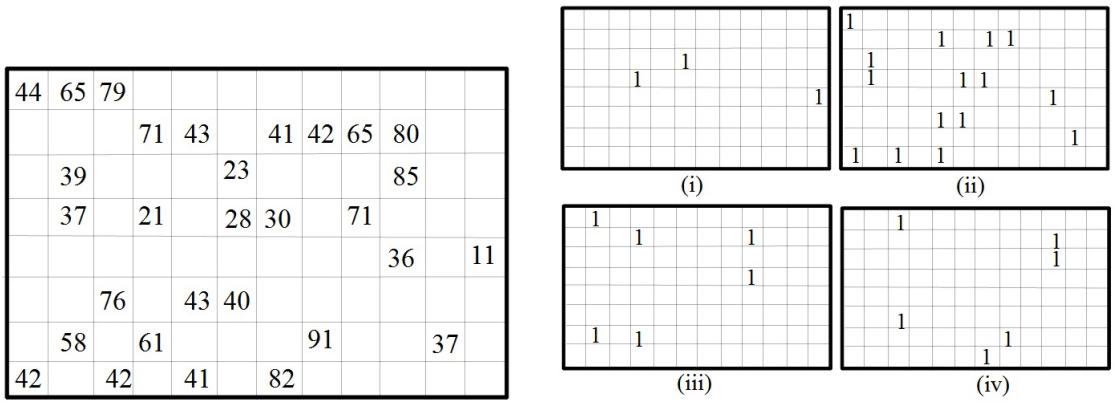
Where $V_1, V_2, .., V_f$ denote the values in the data-space that split it into data ranges, and ϵ

denotes the smallest sensing precision. For example, for a sensed phenomenon whose possible range of values is [1-100], RG_l =[1,25], [26,50], [51,75], [76,100]. Thus, each leaf node needs to report to its $f-1$ siblings, the positions of the nodes of its population conforming to their assigned data range. This can be easily performed at each leaf node by a single scan on the array that is sorted according to sensed values.

3.2.1 Data-space Representation at the Leaf Nodes of the Indexing Structure

Thus far, each of the leaf nodes got hold of the positions of all nodes that are sensing values within its data range(s) of responsibility. In order to create the representation construct $L_{v'_{d,i}}$ of nodes positions for each leaf node $v'_{d,i}$, a bit-map is created. A bit-map is a is a 2D array of a size that maps to the physical region it represents, where each entry represents an area that can be occupied by no more than a single sensor node. Similar to a chess board, a square can be filled no more than one piece at a time. The resolution of the map (i.e. size of each cell) is application dependent. For example, in applications that seek the coverage of a large field, a cell size could be the sensor node communication range.

The map entries/cells are initialized to zero. A leaf node, then, sets the cells occupied by sensor nodes that reported values within its data range. The resulting map can be viewed as a highly sparse 2D array of zeroes and ones, which is then compressed using Run-length coding technique. Figure 5 depicts an example of bit-map construction for the sensed values within one region.



(a) Sensed values in {1, 100} range, reported within a region(blank areas indicate absence of sensor nodes).

(b) Four bitmaps : In (i), all locations that reported values in the range {1, 25} are set to 1 Similarly in (ii), (iii) and (iv) for data ranges {26, 50}, {51, 75}, {76, 100}, respectively.

Figure 5. Illustrative example of the bitmap creation.

3.2.2 Data-space Representation at the Non-Leaf Nodes of the Indexing Structure

In the indexing spanning tree, every two subsequent levels i and j contain, respectively, G_{C_i} and G_{C_j} sets of f sibling nodes, where each set represents a group of neighboring clusters within one region ($j = i + 1$, $C_i = 1, 2, .., f^{d-i-2}$, and $C_j = 1, 2, .., f^{d-i-2}$). On both levels i and j, the data-space of a region is distributed among a group of f sibling nodes.

For each data range R_{G_i}, the corresponding responsible nodes (f in total) at level i compress their maps using run-length encoding and send them to the node in level j responsible for the same data range in the containing region. Upon receiving the f maps for the data range, the recipient node concatenates them according to their geographic locations, which generates one

larger map for the data range of responsibility in the whole region.

In order to provide approximate representation and keep message size fixed across the indexing structure, the concatenation of the set of f maps has to be embodied in a coarser map whose size does not exceed the size of the largest of the f maps. The concatenated map needs to be zoomed out with a scale that reduces its size with a 1/f factor on average.

For example, if we have a geometric area represented in a map of 64 single bit cells, it can determine the presence of up to 64 sensors. At the next level of abstraction, if the scaling factor is 4, this region will be represented with 16 cells, each using 2 bits to specify the number of sensors.

In this fashion, nodes keep approximating maps of different data ranges as they elevate through the indexing structure, providing the ability to supply proper approximations for the data-space. Figure 6 depicts a detailed map construction example and a set of its coarser versions. This data-space abstraction method provides an energy efficient, load balanced, multi-resolution localization tool across the data indexing hierarchy. When an approximate data-space query is received at one of the indexing structure nodes, it can provide an answer identifying the locations of the nodes within its data range of responsibility with a specific level of confidence. A full example for a hierarchical representation of the data-space is shown in Figure 7.

3.2.3 Space Optimized Data-space Abstraction

In the some sensed fields, there are areas within the field that are not covered with sensors because of physical conditions/limitations, or even as part of the coverage plan. In such cases, the representation of these locations within the bitmap becomes an overhead. The elimination

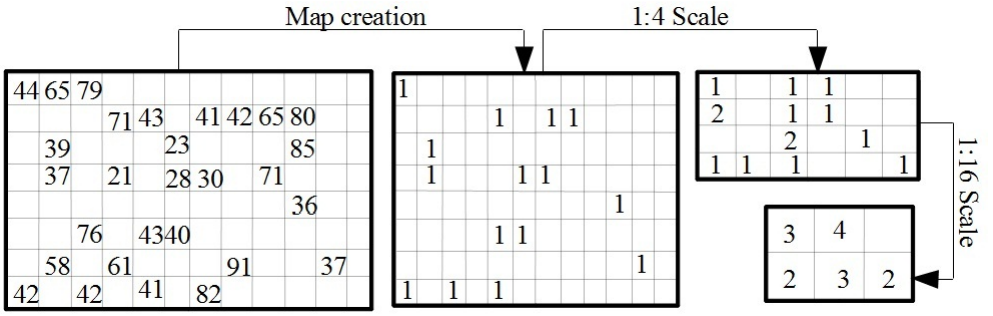


Figure 6. Illustration example for map creation for data range [26-50], and zooming out twice with 1:4 factors.

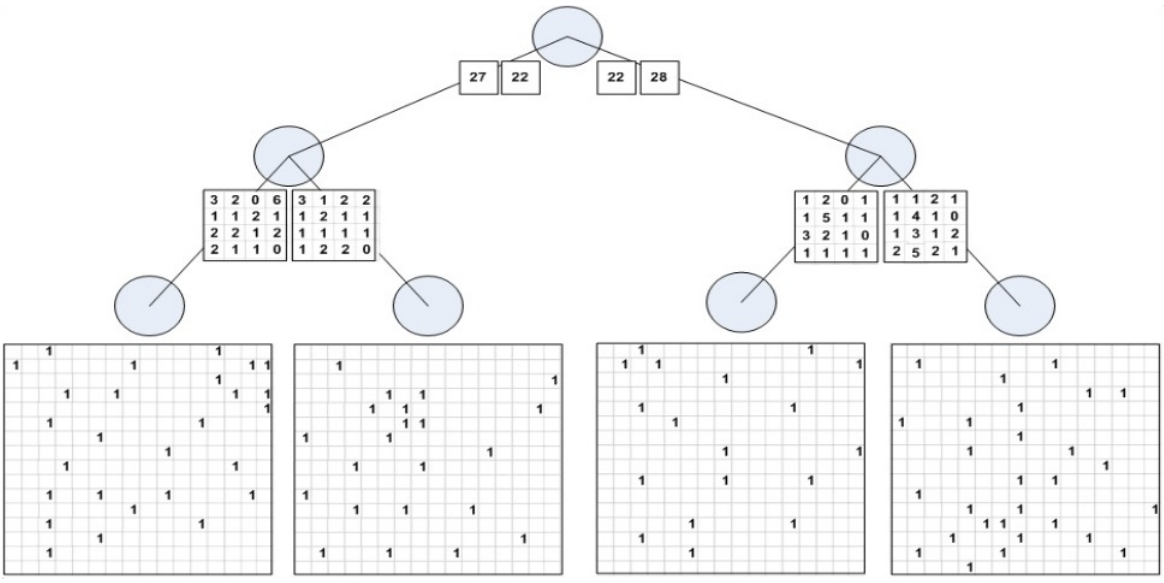


Figure 7. Data-space ($d = 2$, $f = 2$) is split into two ranges. Leaf nodes create maps for their data ranges, and transmit them to the upper level nodes responsible for the same data range. Upper level nodes zoom-out the maps and recursively apply the same process till sink node.

of such overhead can significantly reduce the communication cost of maintaining the data-space abstraction, and hence prolong the network lifetime. Moreover, if the sensor nodes are static within the field, this means that all the locations that do not have sensor nodes can be considered as an overhead. In other words, the sparse bitmaps can be condensed by constructing a map that only represents the locations of existing sensor nodes. Accordingly, this condensed version can be communicated between the data-space indexing nodes, from which the exact map can be reconstructed at the receiving node side. In order for this to be achieved, the receiving nodes need to know the initial distribution of the sensor nodes, regardless of their sensed values. Once this is known, a full map can be simply reconstructed from any condensed version by simply reversing the condensing method. For example, Figure 8 depicts the construction of a condensed version of a region by horizontally scanning the sensor nodes locations and condensing them into a smaller –logical– region, and the corresponding binary representations for its different data ranges. We note that once the condensing step is performed, the data does not need to be represented in a two dimensional form. It can be represented as a single stream of bits marking the locations of corresponding sensor nodes for each map.

3.3 K-D Trees

K-D Trees present a widely used data structure of the category of BSP data structures (5; 6). They are capable of partitioning a given space according to the number of elements populated in the space, in a way that preserves equal count on each side of every partition. Given a set of N sensor nodes, randomly distributed in a 2D plane, at each level of the tree, splits are performed one axis at a time, such that every partition has equal number of nodes. The

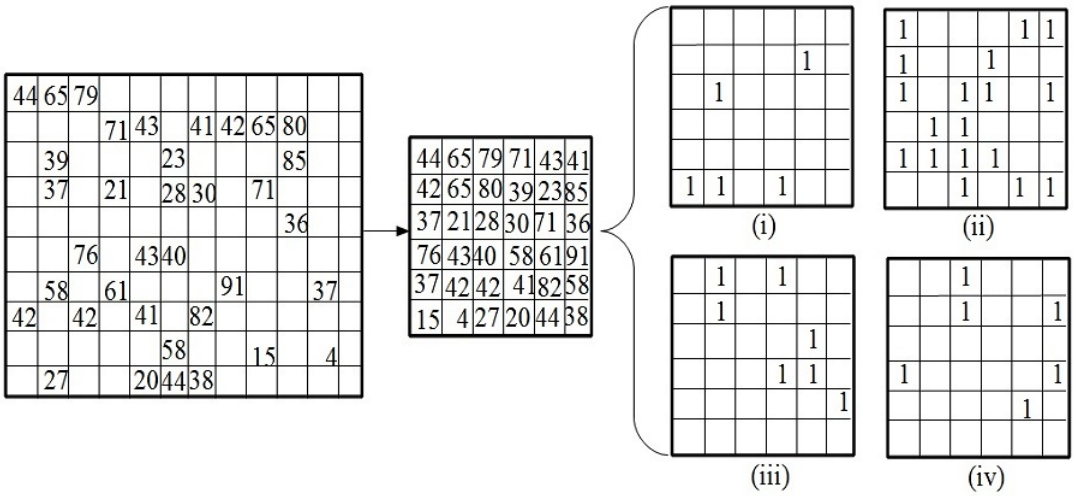


Figure 8. Illustration example for condensed field representation and its corresponding maps for four data ranges [1-25], [26-50], [51-75], [76-100] respectively.

partitioning process is recursively applied, alternating dimensions, till a predefined constant number of sensor nodes is left in every subspace, and that we will call a cluster. The number of recursive partitions to reach this cluster forming is denoted as d. Accordingly, f^{d} clusters will be created, each of which, having a number of sensor nodes no more than a fixed number, denoted by D.

Within each cluster, one sensor node is elected as a cluster head, named as a local cluster head. Similarly, elevating in the partitioning hierarchy, among each set of f neighboring clusters, one sensor node is elected to be the head of this set of clusters, denoted as *level i cluster head*, according to the level of partitioning. This process is applied, till reaching the single sink node which heads the hierarchy of cluster heads. A color-coded description of the elected nodes at

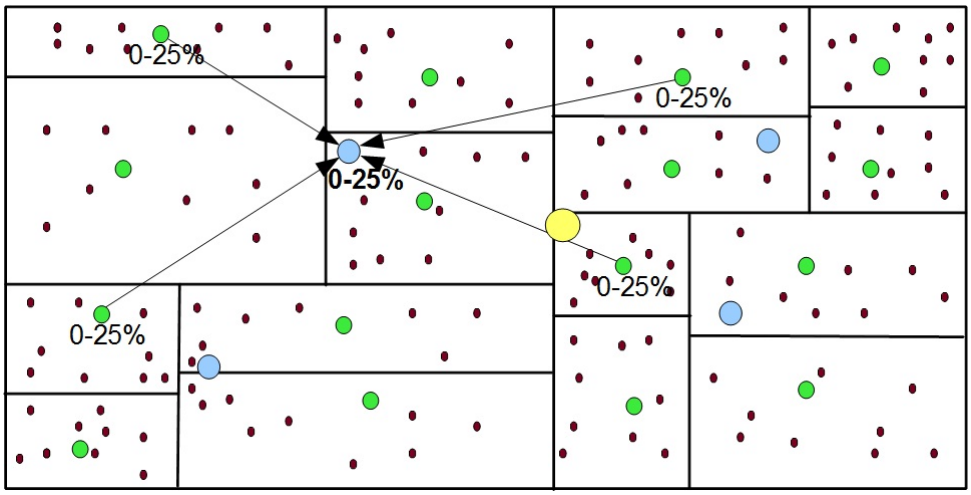


Figure 9. Data-space communication example. For the local cluster heads (in green) responsible for the lowest data range (i.e: 0-25%), maps are transmitted to the upper level cluster head (in blue) responsible for the same data range.

different levels of division of the field was shown in Figure 1.

A spanning tree for the indexing structure is formed as a virtual tree rooted at the sink node. The children of the sink node in the tree are the next level cluster heads. This continues till reaching the local cluster heads which are logically connected to the sensor nodes population. This branching reflects the spatial distribution of cluster heads and the physical sensor nodes. Figure 9 depicts the communication in data-space abstraction for the first data range of the data-space.

3.4 Voronoi Diagrams and Hierarchies

One of the most studied concepts in Computational Geometry (9) is the Voronoi diagram of a collection of points, along with its geometric-dual, Delaunay triangulation. Given n distinct

points in a plane – $P = \{p_1, p_2, .., p_n\}$ contained inside a convex polygon S in $\mathbf{R}$, the bounded Voronoi tessellation $V_{\cap S}(P) = \{V(p_1) \cap S, V(p_2) \cap S, .., V(p_n) \cap S\}$ is defined as the decomposition of the space S into n convex polygons, called Voronoi regions. Each Voronoi region $V(p_i)$ associated with the point p_i, has the property that for an arbitrary point $q(x, y)$; if $q(x, y)$ is inside $V(p_i)$, then *dist(*q, p_i*)* > *dist(*q, p_j*)*, for any other $p_j \in P(i \neq j)$. The function `dist` represents a specified distance measure between two points, which can be as simple as Euclidean distance, or may include other functions that parametrize more context-related information. A specific example is the centroidal Voronoi tessellation CVT is a Voronoi tessellation with the property that the generating point for each Voronoi region is the center of mass of the region.

A bottom-up approach for constructing a hierarchy based on Voronoi cells was presented in (18), where at each level of the hierarchy, a Voronoi tessellation is calculated such that each set of Voronoi cells in a given level is contained in one Voronoi cell in the higher level. However, the boundaries of the Voronoi cells between levels are not guaranteed to fully coincide, which does not result in exact containment of lower level cells in higher levels.

Tree-Maps (19) presented an approach to visualize hierarchical information structures by mapping the full hierarchy onto a rectangular region in a space-filling manner. In (7; 8), Voronoi Treemaps were presented to build a hierarchical spatial data structure that is based on CVTs. Voronoi Treemap is a top-down built data structures, which partitions a given convex polygon space into a set of centroidal Voronoi cells, then recursively partitions each cell into further CVT, given the generating points.

The Voronoi Treemaps represent a useful tool for creating a hierarchical spatial data structure

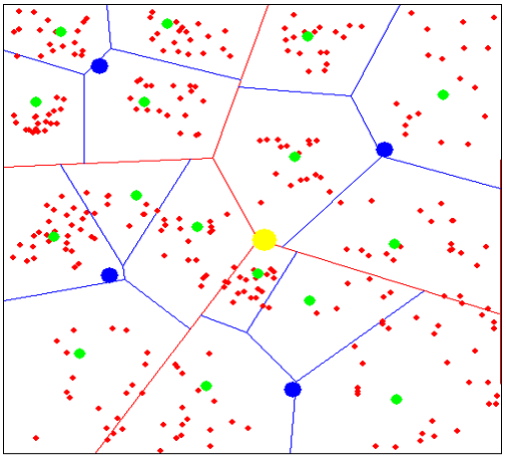


Figure 10. Voronoi Treemap partitioning of 300 sensor nodes. Yellow (largest) node is the sink. The four blue nodes are the next level, and the 16 green nodes are local cluster-heads. Sensor nodes are the tiny red circles.

that splits the given space in multiple granulation levels by creating contiguous non-overlapping regions, and without producing holes. In Algorithm 1, we present a distributed algorithm to create a Voronoi Treemap and deploy the sensor nodes inside a convex polygon bounded field, starting from the sink node. Figure 10 shows an example of a WSN field on which a Voronoi Treemap is constructed.

The presented algorithm runs in time complexity of order $O(log\ n)$, where n is the total number of the Voronoi partitions (i.e, indexing nodes). The message complexity of the algorithm is of linear order $O(n)$. We assume that we are given a distribution function of the phenomena F(region, depth), which returns the location(s) of the point(s) of interest (i.e: hotspots) within this region according to the given depth of the hierarchy. An example of the data space abstraction in a hierarchical Voronoi partitioned field is depicted in Figure 11.

Algorithm 1 Create Voronoi Hierarchy

Input: depth

Output: Creates the parts one level of Voronoi cells in the hierarchy and assigns cluster heads for it. For lowest level, sensor nodes are allocated

```
1: if depth <MaxDepth then
2:    generatingPointsLocations = F(Node.region, depth);
3:    voronoiCells = createCentroidalVoronois(Node.region,
      generatingPointsLocations);
4:    for all cells Vi in the voronoiCells do
5:       assign node C(depth, i) cluster head for Vi;
6:       C(depth, i).region = Vi;
7:       C(depth, i).CreateVoronoiHierarchy(depth + 1);
8:    end for
9: else
10:    // Allocate sensor nodes for each local cluster.
11:    allocateSensorNodesInCluster(Node.address, Node.region);
12: end if
```

3.5 Query Traversal

We note that the higher a given node is in a particular hierarchy, the wider the area for which it is responsible, and the coarser the representation it keeps. When traversing towards the lower levels (at extreme, the terminal/leaf nodes), finer detailed representations are found, albeit for smaller collection for clusters. That is, for a particular node in the hierarchy, detailed constructs of its physical-space representation are found in its child nodes, while detailed/zoomed-in data-space versions are attained at the node's child and nephew(s) which cover the same data range that this node covers. Each leaf node of the tree represents a cluster which contains the exact data of the group of sensors nodes logically connected to the spanning tree through this leaf node.

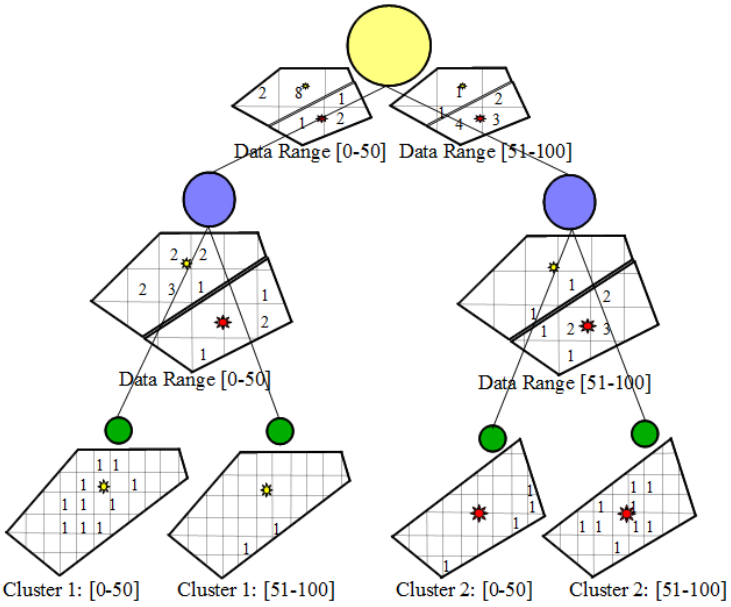


Figure 11. Data-space ($d = 2$, $f = 2$) is split into two ranges. Leaf nodes create maps for their data ranges, and transmit them to the upper level nodes responsible for the same data range. Upper level nodes zoom-out the maps and recursively apply the same process till sink node.

Queries originate at a sink, which we assume is connected to a base station. Upon receiving a query, the root node first checks the query type to decide which representation is inquired. It then analyzes the bounding constraints –if any exists– for range queries, where through the bounding region and bounding data range constraints the node can determine which nodes need to participate in the processing. The solution path is determined step by step, where each indexing node that receives the query checks the intersection of the query's geometric and data range bounds with the geometric area and data range(s) that it covers. The decision of being able to answer the query at each node is taken according to the accuracy requirement of the

query. The behavior of each node is formally specified in Algorithm 2.

Once the query reaches the node(s) capable of answering it with satisfactory accuracy, the response is backtracked through the same path it took from the root node. Each intermediate node waits to receive the response from all the nodes it forwarded the query to. Once received, it combines the query responses. The query results are obtained by concatenation of the data-space maps, while physical-space range queries merge the arrays of sensed values. In the case of extreme values (min or max) the merging trivially preserves the smallest or largest values. See Algorithm 2 - Part 2 for additional details.

Algorithm 2 Query Traversal

Part 1: Query-Forward

Input: Query Q(V, T, C, A)

Output: Immediate Query Response OR A saved memory record to wait for the reception of query response from other nodes

```
1: Receive query(Q(T,C,A));
2: for all constraints C_i in C do
3:   if the constraint cannot be satisfied at this level or by any subtree then
4:     Send back response 'No data available for this query';
5:   else
6:     if accuracy A can be satisfied at this level then
7:       Prepare response and send back;
8:     else
9:       Forward the query to the appropriate node(s) (for physical-space and data-space
         coverage) in the next level (of depth = d + 1);
10:      Keep a record of the query and the number of nodes it was forwarded to, until the
         response(s) come back from the lower level node(s);
11:    end if
12:  end if
13: end for
```

Part 2: Backtrack-response

Input: R(T,Attr[],A, Data[])

Output: Collect all the query responses expected to be received, augment, and forward the result to the node that has sent this query.

```
1: Receive all n expected query responses (R(T,C,A)); //Known from the record saved in
   Query Forward
2: for all attribute i in the Attr[] do
3:   for all response R_j in the received responses do
4:     if T = p then
5:       //Physical-space query
6:       Response_i += Merge Data[R_j];
7:     else
8:       //Data-space query
9:       Response_i = Intersection (Data[R_j]);
10:    end if
11:  end for
12: end for
13: Send back the responses to the node that forwarded this query;
```

CHAPTER 4

MOBILITY MANAGEMENT

4.1 Introduction

Mobile sensor nodes (20; 21) greatly increase the adaptability of the WSNs from the perspectives of: 1. ensuring a level of Quality of Service (QoS) in response to phenomena fluctuation, in the sense of providing better spatial resolution of sampling in desired/targeted areas; and 2. enabling a control over (balancing) the levels of connectivity and coverage. We note that the motion of the nodes may vary in different applications but, from a general perspective, it can be predictable (22), random (23), or controlled (24). For example, in the data coverage problem in WSN (25), controlled mobility of the sensor nodes is utilized in different applications to achieve more efficacious coverage.

An illustrating example of the motivation for mobility handling is shown in Figure 12. The left side of Figure 12, a sensed field with randomly deployed sensor nodes is shown. The right side of the same figure shows the nodes location distribution after the occurrence of an event of interest in the southeast corner of the field. In this case, the application or mobility control algorithm (as (26)) steered more sensor nodes towards that corner, in order to collect more precise information, while still maintaining coverage and network connectivity across the region. Due to this mobility of the nodes required by the application, the underlying distributed indexing structure may become highly skewed, unless it is adjusted to reflect the new distribution of the nodes in

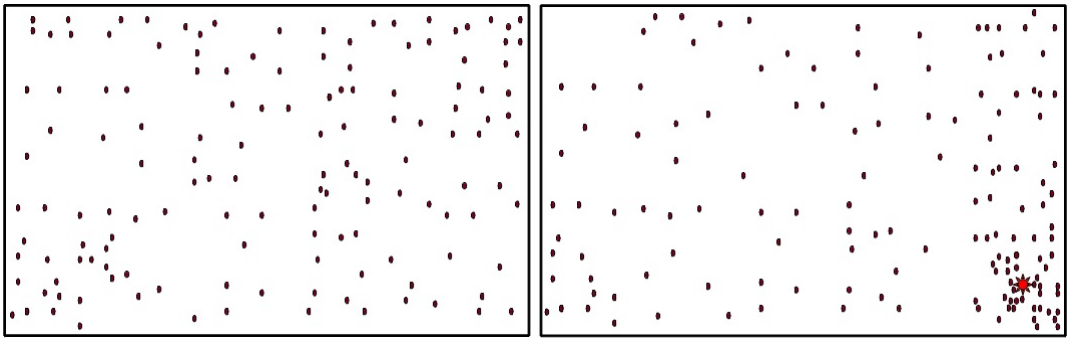


Figure 12. Left Side - a set of sensor nodes randomly deployed. Right Side – nodes distribution after occurrence of an event of interest in the southeast corner.

a balanced way. The main question addressed in this chapter is how to efficiently adapt the indexing structures that manage in-network query processing and aggregation in such mobility scenarios, in response to the change of nodes' distribution, such that the overall maintenance cost is decreased. We emphasize that the actual mobility information as to which nodes should move in what direction is given by the application. Also, it is the application responsibility to guarantee minimum number of nodes needed to provide connectivity and coverage. In order to show our work, we use (26) as the dictating application for mobility.

4.2 Initial Configuration

Assume that logically there are two types of nodes, senor nodes that sense the field and indexing structure nodes that contain the keys to help maintain the indexing structure. Physically, a node can be a sensor node as well as a node in the indexing structure. Further assume that the number of nodes in the indexing structure is n, and the fan-out of each inner node is k, such that the height of the indexing structure is $O(\log_k n)$. The initial setup of the protocol

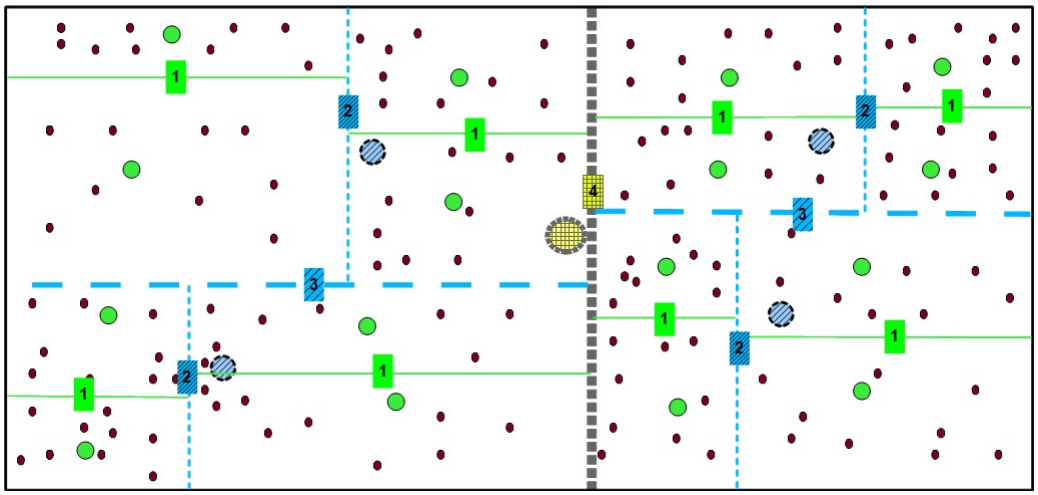


Figure 13. A field with randomly deployed sensor nodes, where the corresponding borders rank are assigned.

assigns an integer rank for each border/hyperplane corresponding to a node in the indexing BSP tree, equal to the depth of the node in the tree (i.e., its level-distance from the root). Figure 13 illustrates the borders rank for a sensed field (color-coded with the same colors according to the splitting order). Each leaf node is responsible for (the sensed values of) a group of m sensor nodes within its vicinity. Sensor nodes periodically (with fixed cycle length) report their sensed values and locations to their respective cluster head.

We reiterate that the motion/displacement of the nodes occurs due to a specific objective (e.g., better coverage due to an observed event in a given geographic region) and, as a result, some leaf node(s) in the indexing structure may find more sensor nodes entering to its vicinity and requesting to join. For example, an event of interest may require more sensor nodes to be moved towards, in order to monitor and report more precise data, as depicted in Figure 14,

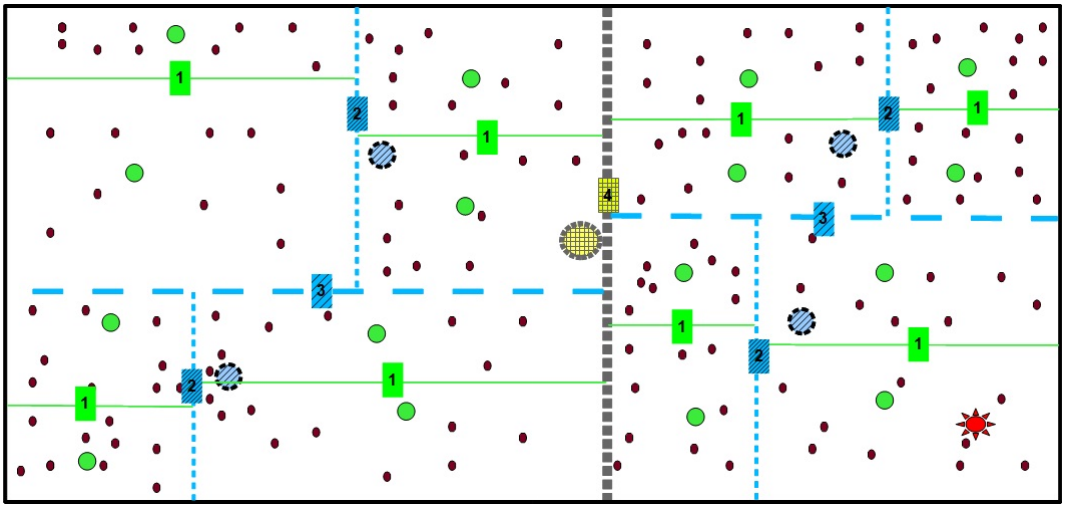


Figure 14. An event of interest in the South-East corner of the sensed field.

which shows a field containing $n = 103$ randomly deployed (small size/red color) sensor nodes. The indexing structure is based on orthogonal bisection (6), performed recursively, such that 16 (thin solid line/green color) local cluster heads are at the first level. Second level of the indexing structure consists of four (thicker dashed line/blue color) intermediate level cluster heads. Last is the (thickest dotted line/yellow color) sink node. Border line shapes follow same nodes drawing/color. In this (initial) configuration, an event of interest is observed in the South-East corner. Also, note that sibling or child/parent node may not be within single hop of each other. In such case, multihop routing of message will be assumed.

4.3 Processing a Request to Incorporate new Mobile Node

Each leaf node has a specified capacity $m' > m$. A leaf node will accept the joining of new sensor nodes coming into its vicinity until reaching the threshold m'. Congestion happens when

a new join request is received at leaf node that has reached its maximum capacity $\mathfrak{m}'$. The leaf node then initiates a request to reduce the size of its space of responsibility by changing the position of one of its surrounding borders/hyperplanes.

The process of border change starts with a communication aiming at changing the spatial splitting locally. The leaf node in the indexing structure experiencing congestion starts by locating the border of its surrounding sides corresponding to the lowest rank convex region. It sends to its sibling node(s) on the other side of the lowest rank border, a change_border_request. When sibling leaf node receives the *change_border_request* message, it starts assessing if it can change the specified border in order to accommodate some of the sensor nodes currently managed by the requesting sibling. The calculation in this case is based on the capacity of the leaf node that received the request. A response is sent back to the requesting node after the calculation. If all the involved leaf nodes have large populations, then they cannot accommodate more incoming sensor nodes, causing them to reject the request. In such case, since the change cannot be handled locally, a new request for changing borders is propagated in the hierarchy to the node corresponding to the next higher rank – i.e., the requesting leaf node sends the request message to its parent node. Upon receiving the request, the parent node checks if the total number of sensor nodes covered by its children is at the capacity limits. If not, it initiates a request to its sibling on the other side of the smallest rank border of its region. The same assessment algorithm runs at the sibling node, which consequently sends the response back. In case of rejection, the same process is recursively applied - in the worst case, reaching the root of the hierarchy (the sink). The algorithm executed locally by the participating node is

formalized in Algorithm 3. Algorithm 4 formalizes the local behavior of the nodes participating in the border-adjustment.

Complexity: In the worst-case scenario, the request needs to be propagated all the way to the sink node. For a BSP indexing tree consisting of n nodes, with a fan-out factor k, at each level, at most $k-1$ request message(s) will be transmitted to change the lowest rank border, and $k-1$ rejection message(s) will be received. In the 2D planar case, $k = 2$ for K-D trees and $k = 4$ if quadtrees are used.

Since, by construction, the height of the BSP with n nodes and fan-out factor k is $\log_k n$, the number of messages required $2 \times (k-1) \times (\log_k n - 1)$, bounding the message complexity of the forwarding stage to $O(\log_k n)$. We note that the overall network-wide running time complexity is the same, since each participating node is executing constant operations to check its current capacity.

4.4 Response Propagation

When a border change decision is taken in non-leaf nodes, all their affected child-nodes are notified, recursively propagating the changes until the affected leaf nodes. Leaf nodes, in turn, inform the affected sensor nodes to change their reporting destination. While this border change information message is flowing through the structure, each recipient node recalculates its population according to the new change to ensure that it is within its capacity. If not, the node finding congestion in its region initiates a new *change_border_request* message and sends it to its sibling node. The important observation is that this particular message is guaranteed to affect borders that are in the sub-tree of the originally changed border, which caused this new

Algorithm 3 Forward Mobility Request

Input: Rank of the border required to change, The count of sensor nodes associated to the requesting indexing node (or its subtree for non-leaf nodes)

Output: A border_change_response OR, in case the whole region is congested, it issues a new border_change_request (if request is received from a child node).

Receive border_change_request (Receiver, Sender.Rank, Sender.nodesCount)

```
1: if Sender.depth == Receiver.depth then
2:   extraNodesCount = Sender.nodesCount - Receiver.optimalNodesCountForCluster;
3:   if Receiver.nodesCount + extraNodesCount <Receiver.maximumNodesCountForCluster
     then
4:     newBorderLocation = calculateNewBorderLocation(Sender.Rank,
       extraNodesCount);
5:     send border_change_response(Sender, accepted, Rank, newBorderLocation);
6:     apply border_change_inform(this, Rank, newBorderLocation);
7:   else
8:     send border_change_response(Sender, rejected, Rank, Receiver.nodeCount);
9:   end if
10: else
11:   Receiver.UpdateNodesCount(Sender, Sender.nodesCount);
12:   if Receiver.nodeCount <Receiver.maximumNodesCountForCluster then
13:     newBorderLocation = calculateNewBorderLocation(Sender.Rank);
14:     send border_change_response(Sender, accepted, Rank, newBorderLocation);
15:     for all childNodes other than Sender do
16:       send border_change_inform(childNode, Rank, newBorderLocation);
17:     end for
18:   else
19:     Rank = Sender.Rank + 1;
20:     send border_change_request (Sibling, Rank, Receiver.nodesCount);
21:     requestingBorderChange = TRUE;
22:   end if
23: end if
```

Algorithm 4 Receive Mobility Response

Input: Rank of the border to be changed, The response (accept or reject), The new border location (in case of acceptance)

Output: Applies the border change for the node, in case of acceptance, Or initiate new request in case of rejection.

Receive border_change_response (Receiver, response, Rank, newBorderLocation)

```
1: if response == accepted then
2:    apply border_change_inform(this, Rank, newBorderLocation);
3:    requestingBorderChange = FALSE:
4: else
5:    Rank = Sender.Rank + 1;
6:    nodeCount = Sender.nodeCount + Receiver.nodesCount;
7:    send border_change_request (Parent, Rank, nodesCount);
8: end if
```

congestion, because the capacity has already been checked/verified at the parent or ancestor node.

The determining of the new border location is based on the population size of the requesting (congested) and responding nodes. For that, we rely on the structural properties of the tree's boundary between the nodes at the same level. Namely, we move the border of the node that has a capacity to incorporate new sensors in a direction perpendicular to the current border's position towards the requesting node position, resulting in shrinking the requesting node's area, and accordingly getting more sensor nodes out of its region towards the accepting node's region. The new border location in the low level requests (i.e, requests between leaf nodes) is determined by the requesting node, which knows exactly the location of all its sensor nodes. In higher level requests, the border location change is proportional to the desired new population

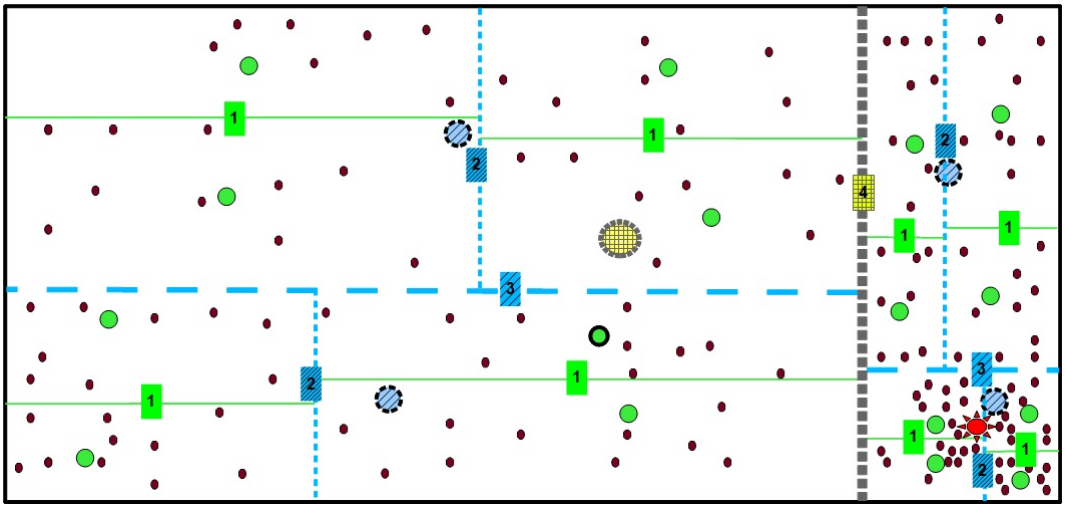


Figure 15. Borders reconfiguration after sensor nodes are moved towards an event of interest in the southeast corner of the field.

size of the congested region. After the change takes place, the node that asked for the border change recalculates its new population to ensure it is within its capacity limits. If not, the node reissues a new *border_change_request*, accordingly. Figure 15 shows the reconfiguration of the borders after sensor nodes have moved towards an event of interest in the southeast corner of the field.

The last step of the protocol involves notifying the mobile motes about the new borders of the tree, so that they know which node-ID to use when reporting the sensed values. This is formalized in Algorithm 5.

Complexity: Algorithm 5 executes when Algorithms 3 and 4 have terminated, and is applied to all the children of the subtree rooted at the node at which Algorithm 4 has terminated. In the worst-case scenario, the execution of Algorthms 3 and 4, will cause the request to be forwarded

Algorithm 5 Apply and Propagate Mobility Response

Input: Rank of the border to be changed, The response (accept or reject), The new border location (in case of acceptance)

Output: Applies the border change for the node, in case of acceptance, Or initiate new request in case of rejection.

Receive border_change_inform (Receiver, Rank, newBorderLocation)

```
1: Receiver.border[Rank] = newBorderLocation;
2: if Receiver.depth == MaximumDepth then
3:    for all sensorNodes do
4:       if sensorNode.Location is out of leaf node new region then
5:          send detach_sensor(sensorNode);
6:       end if
7:    end for
8: else
9:    for all childNodes other than Sender do
10:      send border_change_inform(childNode, Rank, newBorderLocation);
11:   end for
12: end if
```

all the way to the sink node. This, in turn, means that each of the n nodes in the tree will have to be notified about borders change (and, eventually, decide upon the new border's location). Assuming an average of h hops communication between the nodes participating in the tree, the total message-complexity of Algorithm 5 is $O(hn)$. On the other hand, the computation complexity is bounded by $O(\log m)$ – the capacity of each node. Namely, in the worst case, the neighboring nodes (siblings) will have a difference of $m - 1$ motes (assuming at least one mote for a minimal occupancy). Sorting the nodes according to the common-boundary coordinate will take $O(\log m)$, plus the constant time for placing the new boundary.

We note that the mobility scenario that would make the protocol for adjusting the tree incur its maximum cost, is having sensor nodes oscillating around the highest rank border. This case

makes the majority of nodes move towards one side of the border within one update cycle, which causes the indexing nodes to discover congestion and issue *border_change_request(s)*. In the next update cycle, the sensor nodes return back to the other side of the border. In such a scenario, starting from a balanced state, the algorithm behavior would start by a first request at the node(s) adjacent to the highest rank border to change their lowest rank border, which gets accepted at the same level. After the accepting node(s) reach their capacity, while sensor nodes are still crossing the highest rank border towards the adjacent cluster(s), the next request will need to be elevated on level in the indexing tree. On the higher level, the same operation will take place until the managed region is congested.

4.5 Mobility in Voronoi Treemaps

Vornoi diagrams (9) are widely used in WSN applications because of their ability to fit several physical phenomena. Voronoi Treemaps (7; 8) represent a hierarchical spatial partitioning method that is able to provide a hierarchical indexing structure which satisfies Voronoi diagram properties at all of the hierarchy levels. Just as in orthogonal bisection trees (OBT) (5; 6), Voronoi Treemaps are able to manage the sensor nodes in the field, where its leaf nodes are considered as local cluster heads, and the higher level nodes are considered as the intermediate cluster heads till the root (sink) node.

When sensor nodes start moving in the field, because of the occurrence of some events that require redistribution of resources, some local cluster heads suffer from an increasing traffic of sensor nodes coming into their geographical regions. Once this occurs these local cluster heads would want to shrink their geographic region of coverage, in order to keep a number of sensor

nodes within their capacity m'. Unlike OBT, the change of a single edge bordering two –or more– neighboring clusters would violate a fundamental property of Vornoi Treemaps. Thus, in order for a congestion at a specific local cluster to be resolved, and in the best case scenario, the portion of the Treemap between this local cluster head node and its sibling nodes needs to be recalculated. If all the sibling nodes of this congested local cluster head cannot collectively resolve the problem by taking in its excess nodes, the problem has to be elevated to the upper level of the Treemap.

The solution for mobility management in Voronoi Treemaps is of the same essence of the OBT solution, in terms of the requesting, accepting, and response propagation. Algorithms 3, 4 and 5 would apply in the same way to create a request, assess its acceptance criterion, and propagate the response. However, the main difference is related to the action to be taken (i.e, the nature of response). Since Voronoi Treemaps cannot have single edge alterations, and there is a need of recalculating the voronoi diagrams within a region in a way that results shrinking of some cells (i.e, cluster) and expansion of others; Weighted Voronoi Tesselations (27; 28) can be used to recalculate the required part of the Treemap, which can be defined as:

Weighted Voronoi Tessellations: In the basic Voronoi tessellation $V(P)$ it is implicitly assumed that each generator has the same weight. As an extension, a set of parameters W may be given, and to each generator $p_i \in P$ a parameter $w_i \in W$ is assigned. These parameters are the weights. By using weighted generators, it is possible to define weighted distance functions, generating weighted Voronoi tessellations $V(P, W)$.

Once the decision is made at a certain level in the Voronoi Treemap hierarchy for the weights of

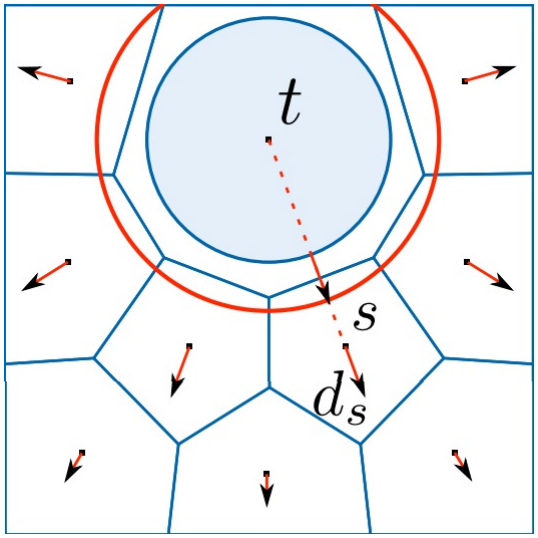


Figure 16. The large growing cell of a site t needs to push surrounding sites s away. To reduce the number of iterations, displacements d_s are determined depending on the distance from t directly rather than alternating between weights and centroid updates for the same effect (30).

its voronoi cells to be changed, the new Weighted Voronoi Tessellation is calculated. Dynamic Voronoi Treemap calculation has been considered as a computationally expensive process (29). In (30), an $O(n \log n)$ iterative algorithm is presented for updating centroidal weighted Voronoi diagrams. Figure 16(30) depicts the update of a cell that is being expanded at the expense of its neighboring clusters area. Since the Voronoi Treemap is a top-bottom structure, the lower levels of all the affected cells (i.e, the child nodes) will have to have their Voronoi Tessellations recalculated, in order to adjust themselves within the bounds of their containing cells.

4.6 Data Indexing Under Mobility

The aim of a in-network data indexing system is to arrange and store the sensed data in a distributed fashion. Indexing tree manages the sensor nodes where each group of sensors report their sensed values and positions to a node of the indexing tree. The recipient indexing nodes store the received information, process them, and elevate approximate constructs across the indexing hierarchy. Mobility causes some of the sensor nodes to move apart from their reporting node(s) of the indexing structure, and hence, get into other node(s) vicinity. This causes unbalance in number of senor nodes reporting to the nodes of the indexing structure. Such unbalance results in the reported data across the indexing structure.

In physical-space abstraction, two approaches can be followed. The first approach is to increase the size of the update message according to the count of the sensor nodes population attached to each node of the indexing structure, in order to keep same sampling distance between the update message values. This would not increase the overall size of physical-space update messages traversed, because the total number of sensor nodes in the field is the same. However, it will create a skew in the size flowing in each branch of the indexing tree, where the larger population branches will have larger size update messages than the other branches. The second approach is keeping the update messages size unchanged, at the expense of increase in the accuracy loss across the indexing hierarchy. In other words, upon receiving a physical-space query, there might be a bigger chance of not being able to satisfy its accuracy requirements from the higher level nodes of the indexing tree, and having to forward the query to next level(s) for achieving the required accuracy. The advantage for physical-space abstraction because of the mobility

handling algorithm is that the change in number of nodes is bounded by the capacity of each leaf node in the indexing tree, $\mathfrak{m}'$.

In data-space abstraction, the change occurring is not because of the motion of sensor nodes, but rather because of the modification of borders location to balance the indexing tree. Due to this change, the bitmap constructs used to represent each data-space are increased/decreased in size, in order to represent the new cluster space. Contrary to the physical-space abstraction, which has its skew factor bounded by the capacity of the indexing structure leaf nodes $\mathfrak{m}'$, the area of a single cluster can increase to approach the size of the whole field. This can only be bounded with the logic of the mobility algorithm, physical constraints of the sensors (i.e., robots moving them), and the field physical barriers. In such extreme case, the large regions can be represented with lower granularity, so the cell size would be coarser than the same level other nodes. This would require high accuracy queries for this region to be forwarded all the way to the leaf nodes. The other solution is to forward the update of such lower level large size cluster(s) as an array of positions rather than a bitmap, and insert them into the bitmap in the higher level node(s) of the indexing tree.

CHAPTER 5

RESOURCE DISTRIBUTION

5.1 Introduction

A common method used in managing the data gathering and aggregation in WSN is to construct a kind of a spatial indexing structure (6), which is maintained/updated in a distributed manner, subject to particular Quality of Service (QoS) constraints (31). Contrary to the centralized settings, in-network coupling of data/information management and indexing structure maintenance involves not only the spatial regions "covered" by a particular node, but also roles/responsibilities of the nodes along the hierarchy of the particular index. Within a given region, the residing group of sensor nodes is assigned the task of monitoring and reporting the sensed values of the phenomena of interest. A particular problem that has been identified and addressed in the literature is how to ensure certain coverage criteria with a given deployment of a set of sensor nodes so that the spatial distribution of the monitored phenomena is matched with a satisfactory accuracy (25; 26). A simple way to define an instance of this problem is how to ensure connectivity and coverage without having any "holes" in the network (32).
Availability of mobile sensor nodes offers an immense flexibility to WSN, in the sense that the location of the sensors can change in order to adapt to certain changes of the values of the sensed phenomenon (21; 33; 20). If a pre-defined event of interest is detected during the monitoring of the sensed field – e.g., a sudden increase in temperature and CO-concentration, indicating a

possibility of a forest fire – then increasing the coverage in that locale may be needed, in order to provide denser coverage and more accurate measurements. However, that process cannot be executed "in isolation" – meaning, completely ignoring the quality of coverage in the rest of the region(s) monitored by the (static and mobile) nodes (34). The left part of Figure 17 depicts a stable scenario of sensor nodes distribution in a given field. The right portion illustrates how some of the sensor nodes (indicated as blank disks with red circular boundary) are selected to-be-moved in new locations inside the region in which an event of interest has been detected, requiring increased coverage.

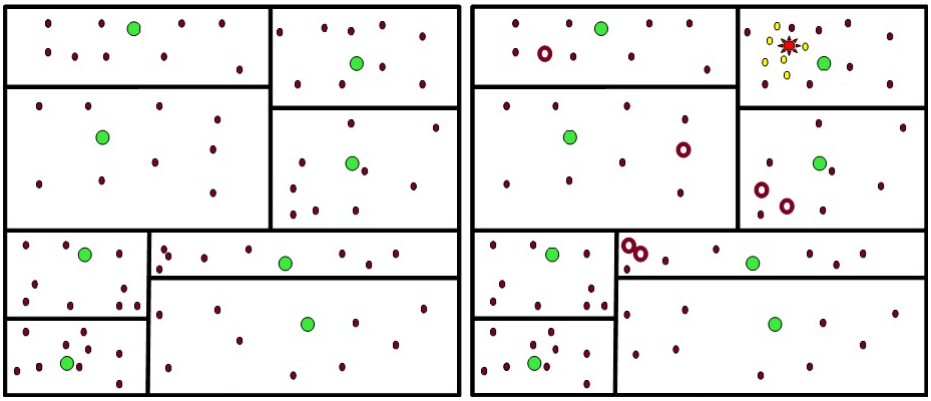


Figure 17. Relocation of sensors in response to an event.

In this chapter, we present efficient distributed algorithms for managing the relocation of mobile sensors upon detection of event of interest in a particular geographic location. The methodology

also caters to the constraint of satisfying the minimum number of nodes required by each of the regions not co-located with events of interest, in order to meet connectivity, coverage, or any other application-dependent demands. Our methodology is able to handle multiple simultaneous requests, and provide the resources for them from the nearest region capable of supplying. The presented methodologies aim at minimizing the communication cost for the "bargaining" process, and seek to supply the resources in a manner that minimizes the total motion, as well as the response time. The algorithm proceeds in a "cascading manner" – meaning, if the regions neighboring the one in which request-generating event is located are not able to satisfy fully the demand, then they provide a partial supply and recursively propagate the request to their (other) neighbors. We assume that the maximum number of nodes required by simultaneous of events in the field is no more than the total number of existing sensor nodes in the field, after satisfying the other regional constraints, i.e, coverage, connectivity, ..etc. Specifically, in this work we also consider the management of such request when a hierarchical structure is present and maintained in a distributed manner (cf. partitions in Figure 17).

5.2 Preliminaries

We assume a sensor network consisting of N nodes $SN = \{sn_1, sn_2, \ldots, sn_N\}$, grouped into geographically collocated K clusters $(C_1, C_2, \ldots, C_K)$. We also assume that under normal initial conditions, each cluster C_i contains $\approx N/K$ motes, which may be of two basic kinds: *static* – S_{C_i}, and *mobile* – M_{C_i} where $S_{C_i} \subseteq SN$ and $M_{C_i} \subseteq SN$, and $SN = \cup_{(i)}(S_{C_i} \cup M_{C_i})$. We assume that the location of the individual nodes are known, either via GPS or via some collaborative trillateration technique (35).

In order to ensure some "desirable" properties – both from the pure networking aspect (e.g., connectivity, coverage), as well as application Quality of Service (QoS) requirements (e.g., density of coverage) – we assume that each cluster has a predefined lower-bound threshold $\Theta_{C_i} \leq (N/K)$ so that $|M_{C_i}| + |S_{C_i}| \geq \Theta_{C_i}$. With this in mind, we note that part of the mobile nodes in each cluster may be "free" to move outside that cluster, without violating the Θ_{C_i} constraint. Hence, $M_{C_i} = M^b_{C_i} \cup M^f_{C_i}$, where $M^b_{C_i}$ denotes the mobile nodes *bound* to C_i and $M^f_{C_i}$ denotes the *free* nodes which can cross the boundaries between neighboring clusters. While the membership of a particular mobile node sn_j may vary – i.e., its state may transfer from *bound* to *free* and vice-versa, we assume that at any time-instant $M^f_{C_i} \cap M^b_{C_i} = \emptyset$.

Each cluster is assumed to have one designated sensor node that will act like a *local cluster-head*, and we use $H(C_i)$ to denote the local cluster-head of the cluster C_i. $H(C_i)$ is in charge of tasks such as gathering and maintaining the information about the status (e.g., locations, expected-lifetime) of cluster's population of nodes; coordinate the operation of the nodes (e.g., increase the sampling frequency); perform analysis/aggregation and information extraction of the measurements from the nodes in the cluster; ..etc. Based on the spatial partitions used, we assume a hierarchy which is constructed from the local cluster-heads, and rooted at a designated *sink*.

For this work, the important responsibilities of a local cluster-head are:

1. *Event detection* – event denotes an occurrence of something of interest (36) and, based on the reported values from the sensors in the cluster, the local cluster-head is in charge

of detecting them[1]. We assume an application-dependent specification of "interesting" events, for which the location of the sensors detecting them is known, along with two values:

- The bounding rectangle – which is, some safety or quality based boundary around the location of an event, approximated by the perimeter of a rectangle.
- The number of sensors needed to be placed around the perimeter of the bounding rectangle – again, an application-dependent parameter. We assume that the sensors will be located at a uniform distance around the boundary.

Hence, we use $\mathsf{E}_{\mathsf{C}_i,j}(\mathsf{L}, \mathsf{B}, n_e)$ to denote that the j-th event E has been detected at location L in the cluster C_i, for which n_e nodes are needed around the perimeter of the rectangle B.

2. *Mobility coordination* – in order to ensure a desired QoS, the local cluster-head may need to direct the mobile sensors toward the location of a given event. We assume the existence of efficient techniques to orchestrate the trajectories of the mobile sensor for the purpose of positioning them at the respective locations around the perimeter of the bounding rectangle which can be viewed as a simplified instance of the techniques in (37) (cf. Sec. 5.4).

There are various spatial partitioning methods (5) and although throughout this work we use rectangular regions, the results can be directly extended to the cases when the field of interest

[1] In this work, we do not consider any composite events.

is convex polygon (subsequently split into a set of non-overlapping regions). A hierarchical spanning tree data structure (indexing tree) is constructed to manage the WSN, which is rooted at the sink node, and has the local-cluster-heads as the leaf nodes. The intermediate nodes are called global-cluster-heads, and we use the term cluster-heads to refer to both local and global-cluster-heads. Various widely used data structures conform to the aforementioned description, and have been used in the existing state-of-the-art indexing systems – e.g., K-D Trees, Octrees (11; 6), and Voronoi Treemaps (7; 8). We note that optimizing the energy consumption due to altering the indexing structure is not considered in this work (i.e., no local cluster-head will ever drop the number of actual sensors in its region below Θ_{C_i}).

5.3 Request

We now proceed with the details of handling the requests for additional mobile nodes, to be made available in a cluster in which an event of interest has been detected.

When a local cluster-head $H(C_i)$ detects an event $E_{C_i,j}(L, B, n_e)$ within its region, it firstly checks whether the mobile nodes from M_{C_i} are sufficient to cover the requirements – i.e., whether $|M_{C_i}| \geq n_e$. If so, the QoS requirements can be satisfied locally. Otherwise, $H(C_i)$ may need to request additional resources.

In the rest of this section, we focus on two basic techniques for handling the request from the local cluster-head that needs more resources to cover an event within its region to the other cluster-heads in the field. First, we present the centralized protocol, followed by two variants of a distributed protocol.

5.3.1 Centralized Requesting

Under the centralized requesting scheme, once a local cluster-head recognizes the need of resources because of the detection of an event in its region, following is the protocol that is executed:

1. $H(C_i)$ sends a request to its parent node in the indexing tree by generating the message *Request*$(H(C_i), r, j)$, the semantics of which is: *The local cluster-head of* C_i *is requesting* r, j *nodes to satisfy the request of servicing the detection of its* $j-th$ *event.* See Figure 18(a).

2. Once sink node receives a particular request, it broadcasts the *RequestSink*$(m_{id}, H(C_i), r, j)$ message to its children, which recursively propagate it down the hierarchy, until it has reached the leaves (recall that leaves are actually the local cluster-heads). The parameter m_{id} is a unique message-ID, in case the sink needs to process multiple requests from the same local cluster-head. See Figure 18(b).

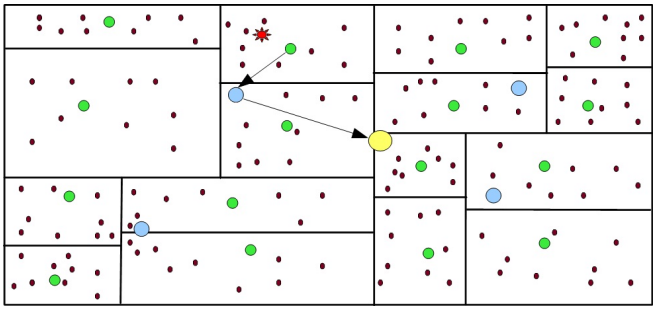
3. Upon receiving the *RequestSink*$(m_{id}, H(C_i), r, j)$ message, each local cluster-head responds with a message containing an information about its free mobile nodes, which could be used to cater the given request. Thus, the local cluster-head of the $l-th$ cluster, $H(C_l)$ will send the message *RequestCater*$(m_{id}, (H(C_l), A_l))$, indicating that it has A_l available nodes. We note that $A_l \leq |M^f_{C_l})|$ (the number of the "free" mobile nodes) since $H(C_l)$ may be processing multiple request (including some due to events in its own geographical region). The *RequestCater*$(m_{id}, H(C_l), A_l)$ message is sent to the parent node in the hierarchy, and each parent aggregates the $(H(C_l), A_l)$ pairs for a corresponding m_{id} before propagating it further up the hierarchy. See Figure 18(c).

4. Once the sink has received the availabilities of individual clusters, it calculates the best manner to move resources in response to *Request*($H(C_i), r, j$), and individual messages are sent down the hierarchy, notifying individual local cluster-heads how many of their available nodes should be forwarded towards C_i. Each local cluster-head $H(C_l)$) whose resources will need to be moved, will receive the message *Allocate*($m_{id}, H(C_i), L(H(C_i)), H(C_l), a_l$), where $a_l \leq A_l$ is the number of nodes to be moved towards $H(C_i)$, located at $L(C_i)$. See Figure 18(d).

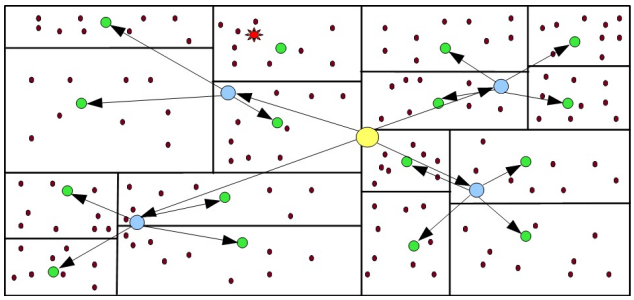
Figure 18 depicts the steps of the centralized requesting protocol. The hierarchical index in this figure is based on a K-D Tree structure (orthogonal bisections). It shows an example scenario prior to the event. We note that the criterion for selecting the resources to be forwarded is discussed in section 5.4.

Assuming an n nodes K-D Tree as an example indexing structure, the complexity of executing the centralized protocol can be characterized as follows:

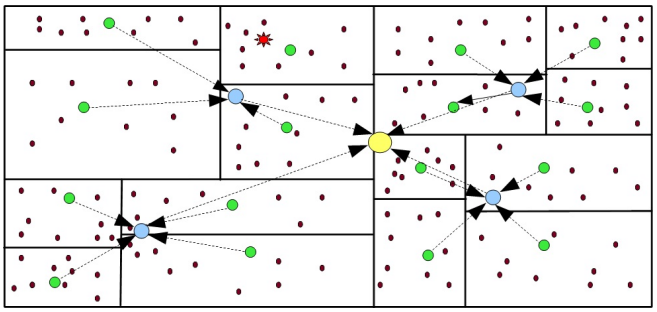
- $\log n$ messages are needed to propagate the request from the $H(C_i)$ to the sink node.
- $n/2 = (O(n))$ messages to send the information request from the sink back to all the local cluster-heads. However, since some of them may be transmitted in parallel in different sub-trees, the time, in terms of hops, is bounded by $O(\log n)$.
- $O(n)$ messages which are from the local cluster-heads towards the sink – again, bounded by $O(\log n)$ in terms of time (although two children will need sequential transmission towards their parent, the bound is still $O(\log n)$).



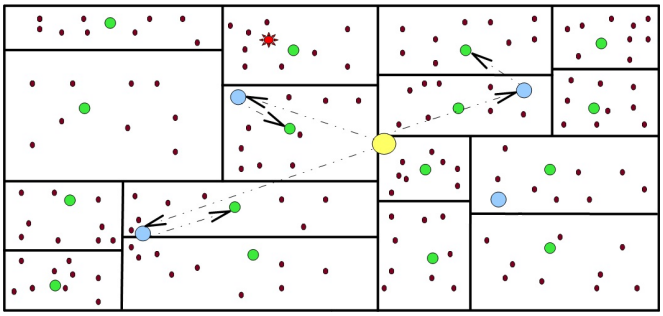
(a) Step 1: The local-cluster-head (Green) detecting the event sends a request to its parent node (Blue), which is forwarded to the sink node (Yellow).



(b) Step 2: The sink node (Yellow) requests information about available resources from all the local-cluster-heads (Green) through the data structure hierarchy.



(c) Step 3: Available resources information is sent to the sink node (Yellow) through the data structure hierarchy.



(d) Step 4: Sink node (Yellow) sends out the decision about resource forwarding to the involved local-cluster-head (Green) nodes throughout the data structure hierarchy.

Figure 18. The centralized requesting process in a sensed field spatially split with orthogonal bisection, with a K-D Tree indexing structure. The middle large yellow node represent the sink node, the four blue medium nodes are the global-cluster-heads, the sixteen green small nodes are the local-cluster-heads, and the tiny red nodes represent the sensor nodes distributed in the field

- Let k denote the number of local cluster-heads selected to participate in sharing their resources with $H(C_i)$ ($k \leq n/2$).

Thus, the overall communication complexity (message cost) is bounded by $O(n)$, in terms of number of messages, and $O(\log n)$ in terms of time.

5.3.2 Distributed Request Management

The distributed structural requesting protocol aims at minimizing the overall communication cost via exploiting spatial locality. We now present two approaches for handling a request for additional resources. The first one, called *structure-based* is relying solely on an existing hierarchical index structure, whereas the second one proposes a coupling between the indexing structure and geographical proximity.

5.3.2.1 Structure-Based Distributed Request (SBDR) Management

Assuming a Binary Space Partitioning (BSP) structure which has recursively divided a given space into contiguous non-overlapping regions, each border/hyperplane corresponding to node in the indexing BSP tree, has a unique *level* (i.e., distance from the root).
When $E_{C_i,j}(L, B, n_e)$ is detected and $H(C_i)$ determines that additional sensors are needed to satisfy the QoS criteria, the SBDR protocol proceeds as follows:

1. $H(C_i)$ sends the message *Request*$(H(C_i), r, j)$ requesting additional mobile nodes to *its sibling node(s) in the indexing tree* which is sharing a common border and parent. See Figure 19(a).

2. In the case that $H(C_{s,i})$, the sibling of $H(C_i)$, can cater to the request, it responds with *Granted*$(H(C_i), j, r_s)$. Clearly, for this we need that $r_s > r$, and the nodes are selected from $M^f_{C_{si}}$ which is properly updated.

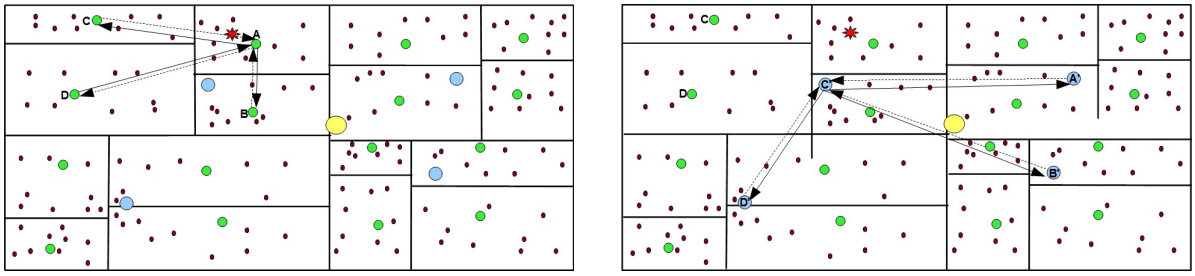
3. In the case that $H(C_{s,i})$ cannot cater to the request, it will send the message *Deny*$(H(C_i), j, r_s)$. The meaning is that, although it cannot fully grant the request, the sibling is still able to provide $r_s \geq 0$ nodes. In this case, $H(C_i)$ will forward *Request*$(H(C_i), r - r_s, j)$ to its parent.

4. The parent-node of $H(C_i)$, in turn, instead of propagating the request towards the sink, will actually forward the *Request*$(H(C_i), r - r_s, j)$ to its own sibling at the same level and sharing a common border whether it can cater to $H(C_i)$'s request. See Figure 19(b).

5. The procedure is repeated recursively until, in the worst case, the request has reached the root.

The SBDR protocol is illustrated in Figure 19. It depicts the chaining of the messages at two levels from the root, since the request cannot be satisfied at the first level – i.e, by sibling local cluster-heads.

Clearly, both the communication cost and the time for detecting the fulfillment of a particular request will vary for the SBDR protocol. We note that, in the worst-case scenario, the request needs to be propagated all the way to the sink node. Worse yet, the attempts to resolve it locally constitute additional overhead in terms of the time needed to determine the servicing of the request. However, as our experiments demonstrate, SBDR protocol does provide improvements over the centralized protocol.



(a) Local-cluster-head A detects an event, and sends a resource request to its sibling local-cluster-heads B, C and D, which send back the response.

(b) When the request is not satisfied at the local-cluster-heads level, global-cluster-head C' checks the resource availability with its sibling nodes (A', B' & D')

Figure 19. SBDP with a K-D Tree indexing structure. The middle large yellow node represents the sink node, the four blue medium nodes are the global-cluster-heads, the sixteen green small nodes are the local-cluster-heads, and the tiny red nodes represent the sensor nodes distributed in the field

5.3.2.2 Structure and Proximity based Distributed Request (SPDR) Management

The objective of the SPDR variant is to decrease the overhead induced by the sibling-to-parent communication in the SBDR protocol. We observe that some local cluster-heads which are not siblings may still share a common border. To capitalize on this, in addition to the sensor nodes physically belonging to its cluster, each local cluster-head will maintain a list of its "cousins" – which is, the sibling node and the nodes sharing common border.

Upon detecting an event $E_{C_i,j}(L, B, n_e)$, the local cluster-head $H(C_i)$ executing CPDR protocol initiates the following:

1. $H(C_i)$ sends the message *Request*$(H(C_i), r, j)$ requesting additional mobile nodes to *its geographically neighboring nodes with which it is sharing a border*. See Figure 20(a).

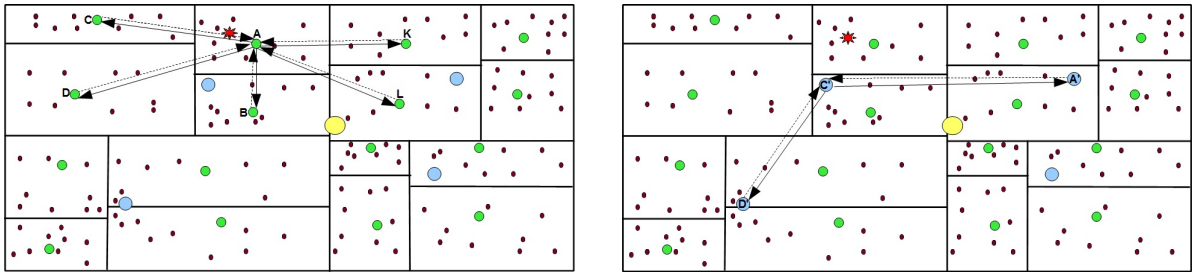
2. The sibling node $H(C_{s,i})$ and each of the Boarder-Neighbors ($BN(H(C_i))$ who can cater to the request, responds with *Granted*$(H(C_i), j, r_s)$. In this case, the request is no longer propagated. $H(C_i)$ notifies its sibling and its neighbors how many mobile nodes each of them should dispatch.

3. If the sibling node and some of the $BN(H(C_i))$ cannot cater to the request, they will each send *Deny*$(H(C_i), j, r_s)$. Note, however that, unlike the SBDR protocol, now the sum of the r_s values from the sibling and the neighbors combined, may actually satisfy the request.

4. If not, the message *Request*$(H(C_i), r - \Sigma(r_s), j)$ is propagated to the parent of $H(C_i)$, and parent recursively repeats the procedure. See Figure 20(b).

Figure 20 shows the messaging at two different levels in the hierarchy, where the request cannot be satisfied at the first level, i.e, through sibling local-cluster-heads communication.

Again, we note that the worst-case scenario in terms of the upper-bound is the same as the centralized protocol – and, once again, in the worst case scenario we have the additional overheads of the attempts to resolve the request locally. However, in practice, one can obtain improvements – as demonstrated by our experiments.



(a) Local-cluster-head A detects an event, and sends a resource request to its neighboring local-cluster-heads (B, C, D, K & L), which send back the responses.

(b) When the request is not satisfied at the local-cluster-heads level, global-cluster-head C' checks the resource availability with the neighboring global-cluster-heads (A' & D')

Figure 20. SPDR with a K-D Tree indexing structure. The middle large yellow node represent the sink node, the four blue medium nodes are the global-cluster-heads, the sixteen green small nodes are the local-cluster-heads, and the tiny red nodes represent the sensor nodes distributed in the field

5.4 Supply

In this section we present the methodology of fulfilling a resource request, starting with the process of acceptance of requests, selection of the nodes to be moved and moving them towards the cluster which has signaled a request.

5.4.1 Strategy of Acceptance

Upon receiving a resource_request, a cluster-head node compares the number of requested sensor nodes R_i to the number of available resources within its region V_i. If the available resources are sufficient to cater for the request, i.e, $V_i > R_i$, an acceptance message is sent to the requesting node, and the process of selecting the sensor nodes to be moved and moving them starts immediately.

Contrarily, if the available resources are less than the required resource, i.e, $V_i < R_i$, the request cannot be rejected. The reason for this, is that in a global view of the field, sometimes no single cluster might be able to suffice the needs for one request. However, a set of clusters can provide a number of the needed resources, which make them collectively able to suffice the new event needs. Therefore, in such scenario, the cluster-head receiving the request sends back a partial_acceptance message, indicating that it will be able to provide V_i resources. Accordingly, when the requesting node receives this message, it forwards the request to another cluster-head node –according to the requesting strategy– with the required number of resources updated, i.e, $R_i = R_i - V_i$. Simultaneously, the partially accepting cluster-head node will start forwarding the V_i sensor nodes, which will create "some" sufficiency for the requesting cluster until further resources arrive. In other words, because of the partial acceptance feature, each request is –most likely– going to add some help to the requesting node, unless the whole region is starving.

5.4.2 Nodes Selection (Which nodes to move?)

Once a local-cluster-head sends the requested resources –or some of them– to the requesting local-cluster-head, a criterion is needed to determine which specific sensor nodes are the ones to be forwarded. The selection criterion may involve the following metrics:

- Speed of Arrival/Travel Distance: The cluster-head node selects the sensor nodes to be moved such that they would arrive to the destination in the fastest way (or travel the shortest distance). This selection would vary according to the motion path (i.e, Manhattan, direct straight path, ...etc.).

- Local Configuration Balance: Maintaining the balance of the sensor nodes distribution inside the accepting cluster. Accordingly, the cluster-head selects the nodes to move in a way that minimizes –or better eliminates– the need of moving the remaining nodes inside the cluster to maintain its internal constraints (i.e, connectivity, coverage, ..etc).
- Global Configuration Balance: Maintaining balance of the sensor nodes distribution in the whole field. The goal of this balance is to keep the available of the M_C mobile sensor nodes distributed across the field, which helps having resources available near to possible future events. This also balances the load on the indexing tree, which keeps –relatively– equal load of network information updates on the tree branches. We note that this option is possible in a straight forward fashion in the centralized solution. However, including this metric in distributed techniques would incur added overhead.
- Energy Consumption: The cluster-head node selects the sensor nodes to be moved according to an optimization function which minimizes the consumed energy. The optimization can focus on the energy consumed for communication, or the energy consumed in motion, or a weighted factor of each of them.

5.4.3 Movement strategy (How to move the nodes?)

The sensor nodes selected to be moved towards the requesting cluster are informed by their local-cluster-head. The supply of sensor nodes to the requesting cluster can follow different methods. In this subsection, we present two methods to supply the requested resources from the –partially or fully– accepting local-cluster-head(s) towards the requesting cluster, then we

follow with a discussion on handling the request inside local-cluster-heads. In the scope of this work, we assume an obstacle-free field, or that obstacle avoidance is implicitly taken care of.

5.4.3.1 Direct Forwarding

In direct forwarding, the sensor nodes move directly towards the requesting cluster. The motion can be in a straight path or Manhattan, depending on the application setup. Once the nodes are decided to move, they are informed by their local-cluster-head, and given the location of the cluster of destination. The nodes leave their cluster towards the destination cluster. On their way to the destination, the sensor nodes can turn off their sensing devices and radio transceivers until they arrive, where, when the sensor nodes pass through intermediate clusters, they do not need to report information. For some data intensive applications, the passing sensor nodes can turn on the sensing and reporting, depending on the speed of motion and the distance traversed inside the cluster. Figure 21 shows the direct forwarding of sensor nodes towards the cluster containing the event.

5.4.3.2 Relayed Motion

The relayed motion depend on setting up the path of motion of the senor nodes through the intermediate clusters before starting the real motion. The goal of this type of motion is to minimize the traveled distance by each sensor node, and provide faster supply to the new events, especially when the available resources for supply are more than one cluster away, i.e, not direct neighboring. The method starts once resources are decided to be moved from cluster C_{source} to cluster C_{dest} passing, in sequence, through clusters C_i, where $i = 1, 2, .., k$. In the path setup, each local-cluster-head is informed with the local-cluster-head before it in the sequence, the one

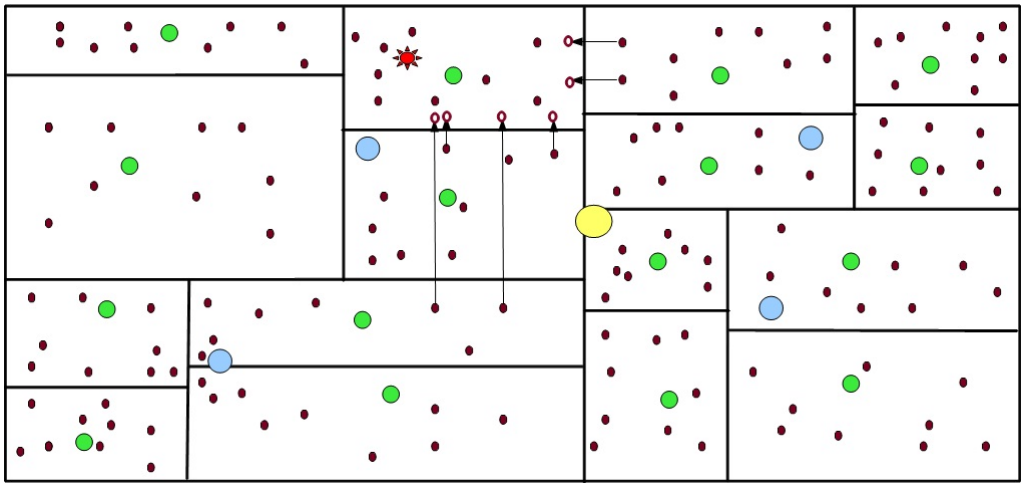


Figure 21. Six sensor nodes moved towards the cluster containing the event coming from three different supplying clusters.

after it, and the number of resources to be supplied. After all the local-cluster-heads in the path are informed –and possibly requested to confirm, for some applications– the real motion starts. Each cluster-head sends the required amount of resources to the next cluster in the sequence, starting from C_{source} through C_i to C_{dest}. This method gives the advantage of faster delivery of the sensor nodes to the destination, regardless of the travel distance. However, it is at the cost of more unbalance during the transient period, where some intermediate clusters may have less number of nodes than its minimum requirements. Also, all the cluster head nodes along the path need to know that they are participating in this scenario. Figure 22 shows the relayed motion of sensor nodes towards the cluster containing the event, passing through intermediate clusters.

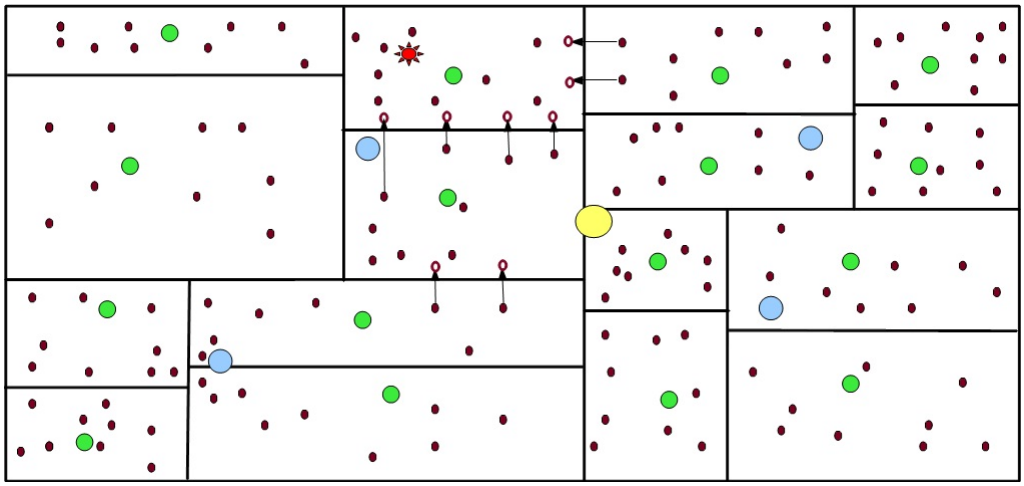


Figure 22. Six sensor nodes moved towards the cluster containing the event coming from two supplying clusters, where one of them (the bottom cluster) relays two more sensor nodes from its population for the cluster beneath it. It accordingly receives other two sensor nodes, which it can place at the appropriate positions inside the cluster.

5.4.3.3 Intra-Cluster Motion

Once the sensor nodes from other clusters have reached the one who has requested resources, the local cluster-head will need to execute a re-allocation algorithm. As mentioned in Section 5.2, in this work we assume that an event is associated with a bonding rectangle, and each type of an event has distribution of locations for the sensors around the boundary.

With this in mind, the re-location inside a given cluster can be readily accomplished using the heuristics from (37). We note that there are different variants of the re-location problem – e.g.: minimize the latest arrival time; determine locations that will maximize the reachability/coverage, and with a given time-budget, ..etc. (26). In our work, we assume the simplest variant – minimizing the latest arrival time, with the known destinations' location. This is

illustrated in Figure 23 – showing a zoomed-version of the cluster in which an event has been detected in Figure 22. As can be seen, some of the previously available sensors, along with the newly-arrived ones, are routed towards the predetermined locations along the perimeter of the rectangle bounding the event.

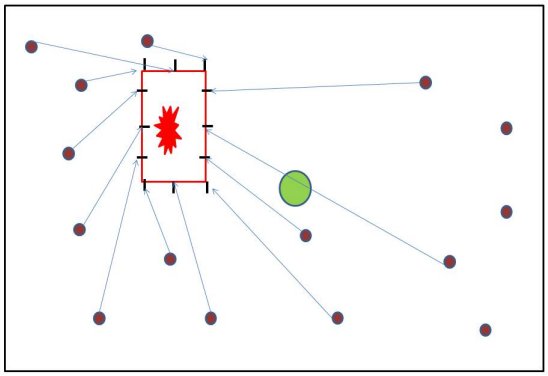


Figure 23. An example of intra-cluster movement of nodes.

CHAPTER 6

EXPERIMENTAL RESULTS

The presented abstraction system was simulated using the SIDnet-SWANS WSN simulator (13) based on Jist-SWANS discrete event simulation engine (14), as a 500 nodes network randomly deployed in a square field of 300 meters length The simulated nodes apply MAC802.15.4 protocol for MAC layer, and Shortest Geographical Path Routing for routing layer. The power consumption characteristics are based on Mica2 Motes specifications, MPR500CA. Each sensor node has a GPS to obtain the location information. The Different types of data distributions were considered to simulate sensing fields of different phenomena.

We execute three types of queries on the simulated network:

1. Physical-space queries – asking about the sensed data of a specific geographic area (i.e: ***Q(v, G, A)***).
2. Data-space queries – asking about the position of nodes sensing some specific data range (i.e: ***Q(l, R, A)***).
3. Hybrid queries – merging the first two types together by creating a physical space query over a bounded data range (i.e: ***Q(v, G, R, A)*** or ***Q(l, G, R, A)***).

6.1 Static Data Indexing

This section starts with a comparative evaluation for the abstraction methods against the current state of the art, followed by presenting the energy cost and query latency for the indexing

system in a static WSN. We compare the physical-space abstraction method to the Gaussian models approximation method presented by Meliou et. al (15). For the data-space abstraction however, to our best of knowledge, there are no available representation models to compare experimental results to in the WSN literature.

6.1.1 Abstraction Techniques

In order to evaluate our abstraction techniques, we present the resulting errors from them against the state of the art. The approximation error calculated for the presented physical-space abstraction, and the Gaussian approximation method in (15) is based on normalized root mean square error (NRMSE), In the case of (15) we report the average error and for the case of regular sampling in our method we report the interpolation error of the estimated values.

In Figure 24, the results for the normal distribution show that the Gaussian method starts with less accuracy than the presented sampling method, but ends up achieving a better precision on the highest level. This is intuitively reasonable, as the global data distribution of the underlying field tends to follow a normal distribution, fitting well with the abstraction method of (15). Nonetheless, because the data in different regions of the fields may follow different distribution functions (cf. (38; 39; 40)), the method used in (15) fails to capture the data with the same efficiency as our presented method.

The results of the other distributions in Figure 24 show that the sampling method achieves better accuracy, ranging from 10% to 90%. It is also important to mention that the average error comparison does not capture an important feature many sensor networks applications require, i.e. querying maximum and minimum values.

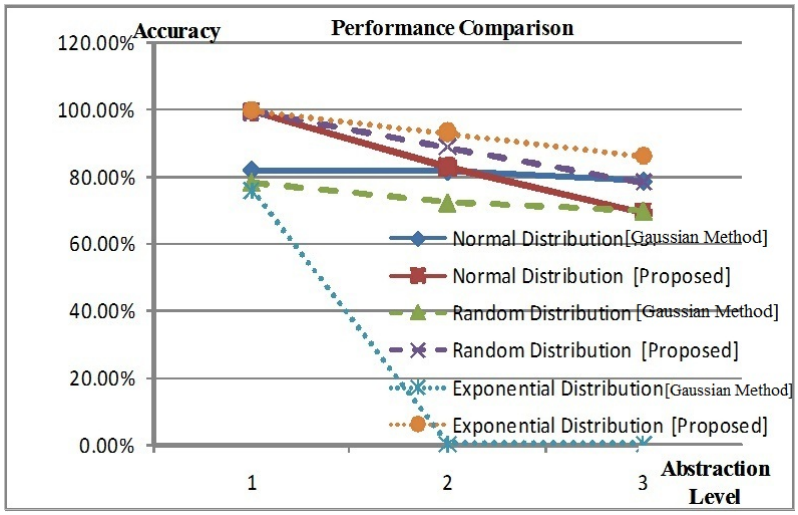


Figure 24. Approximation error comparison (Gaussian vs. sampling) for normal, uniform, and exponential distributions.

Figure 25 depicts the effect of changing the sample size on the abstraction error. The results show clearly that the increase of the sample size reduces the average abstraction error. This reduction varies with the different data distributions. For example, in the normal distribution, the results show that the increase of sample size with more than three sample elements would not result in any further reduction of the abstraction error. On the other side, the increase of sample size for random and exponential distributions reduces the abstraction error.

6.1.2 Data Indexing and Query Processing

The presented system has shown good performance in terms of communication cost and latency for a wide variety of queries. Physical-space, data-space, and hybrid queries were applied to the system with different levels of accuracy, and bounding constraints on geometric field,

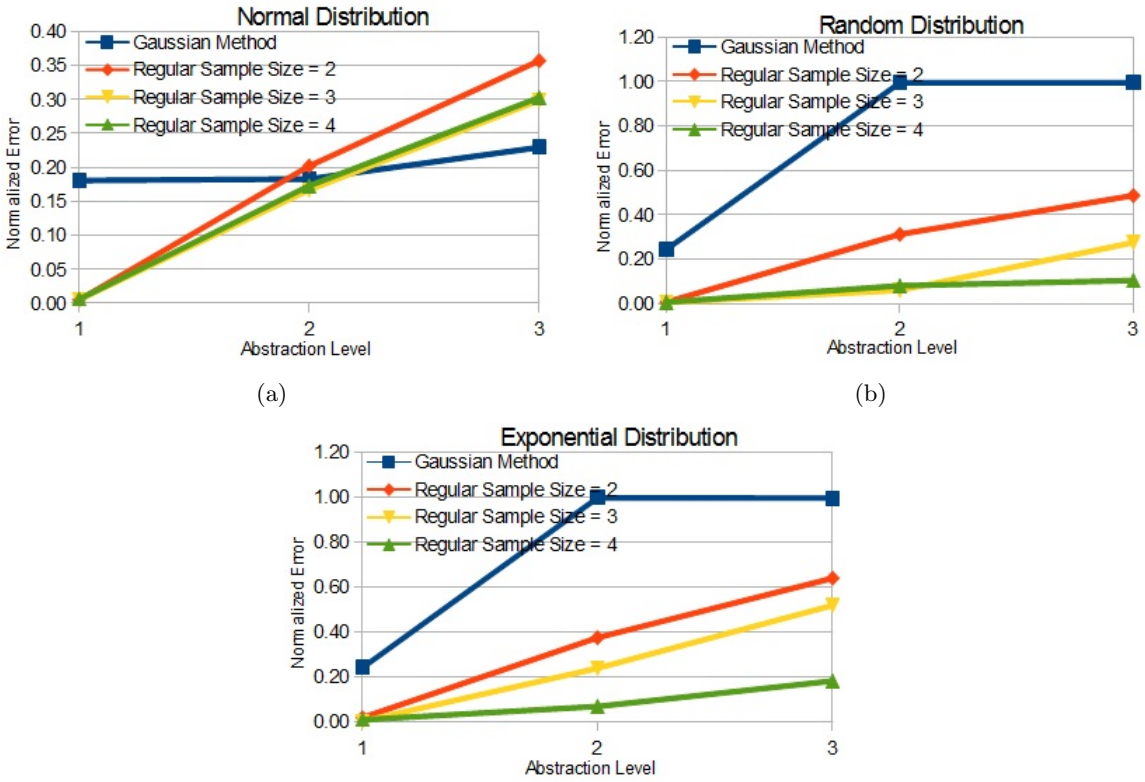


Figure 25. Approximation error comparison (Gaussian vs. sampling) for different distributions with varying the sample size.

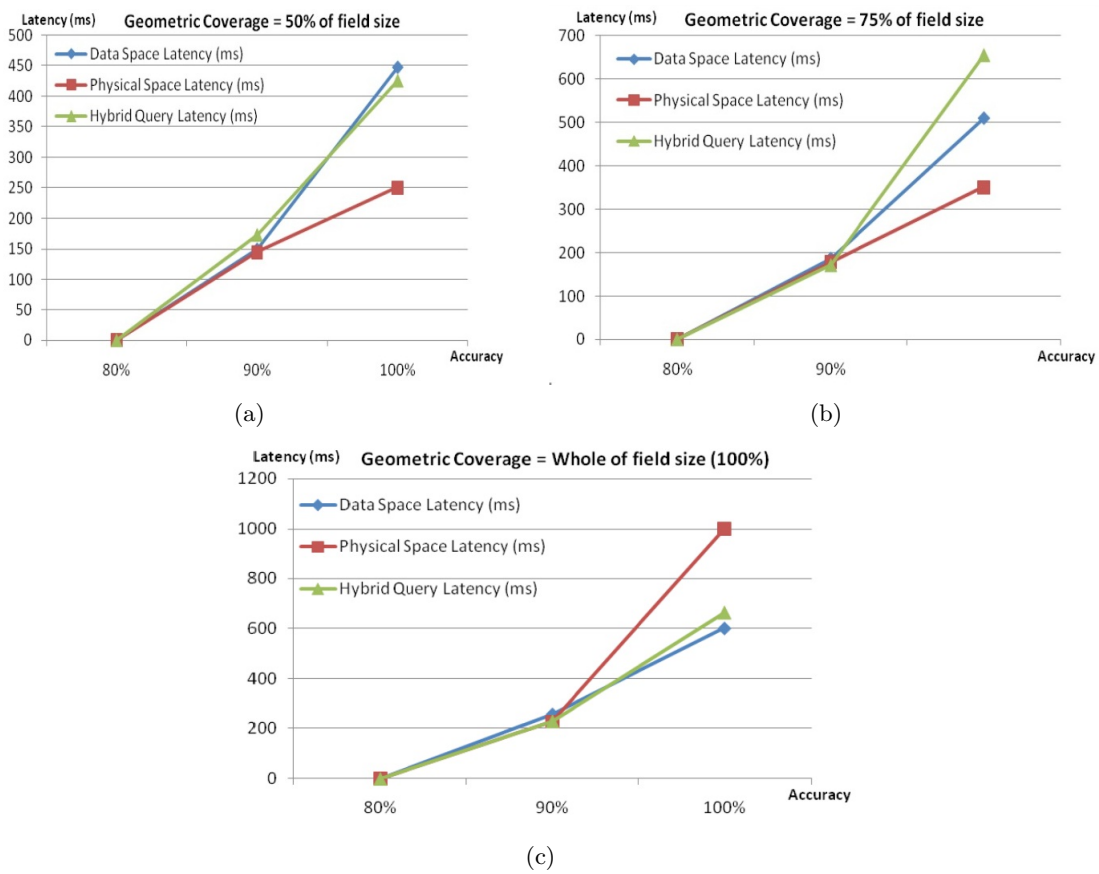
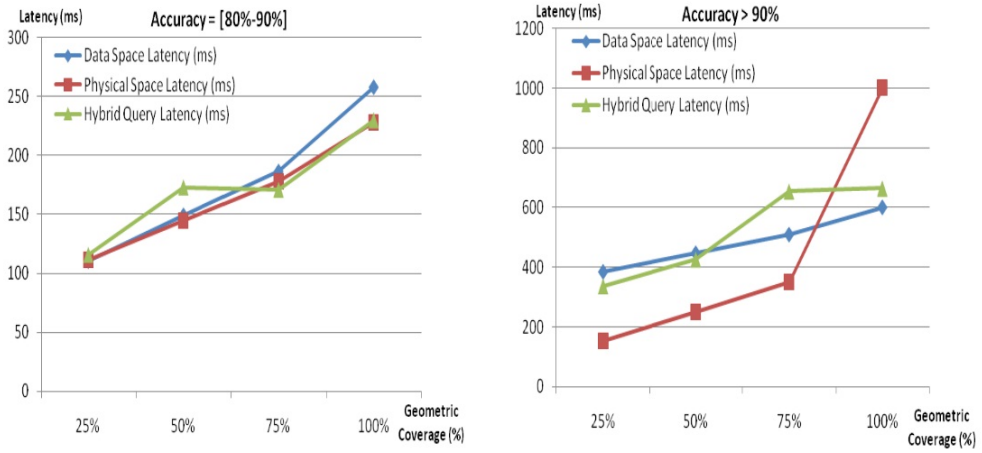


Figure 26. Latency Vs. Accuracy corresponding to the geometric coverage of query as percentile of the area of the sensed field.

and data ranges.

The single-attribute-query results, depicted in Figure 26 and Figure 27, show the change in the latency of query response with the change of desired accuracy, and geometric bounds (represented as percentile of the field size). In Figure 28, the communication cost is shown for the cases in Figure 26 and Figure 27.



(a) (b)

Figure 27. Latency Vs. Query Coverage plot for accuracy = [80%-90%] & [90%-100%].

The results of queries involving single attribute constraint show linear reduction in the number of messages required for response, according to the specified coverage and accuracy. Such linear reduction reflects the reduction in communication cost for querying. Query latency also varies from immediate (zero sec. latency) approximate response at sink node up to about one second to provide an exact answer (i.e. accuracy = 100%) for a query inquiring data about the whole field. This maximum latency (one sec) is the baseline to which the analysis of the abstraction method's latency results has been compared.

The query latency results for the three types of queries in Figure 26 and Figure 27 show that the data-space queries have higher latency than the physical space queries when there is no full query coverage. This is due to the nature of locality of the indexing structure nodes for the physical-space abstraction compared to the data-space abstraction. For a physical-space query

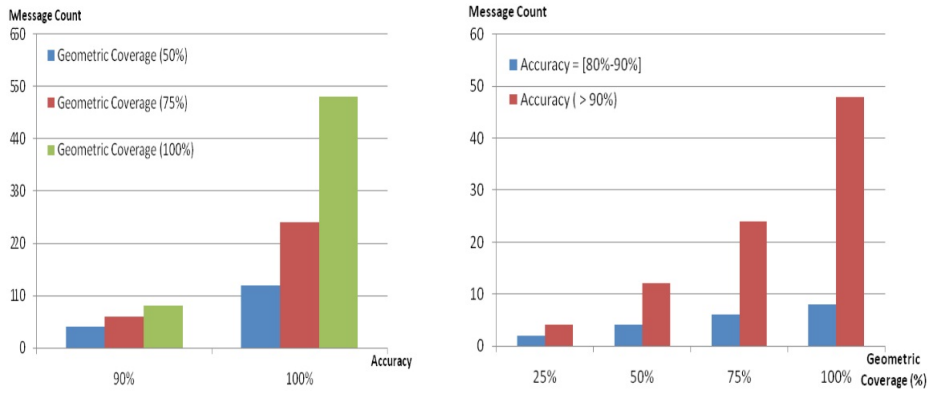
if a node doesn't satisfy the desired accuracy, it forwards the query to its child node(s), while for a data-space query the node may send the query –according to the data range– to either its child node or to its nephew(s) which is intuitively farther than all its child nodes because of the spatial partitioning.

In Figure 29, simulation results are shown for queries containing query coverage constraints on multiple attributes which represent four simulated phenomena. The first parts ((a), (b), and (c)) show the case of identical constraints for all queries, which means that it is similar to a replica of multiple single attribute queries. In this case we see a latency increase of 200ms for all the query types that is consistent over different levels of regional coverage (50%, 75%, and 100%). This shows that using a unified organized information system achieves communication efficiency without increasing latency overhead.

The second part of Figure 29 ((d) and (e)) depicts the results for the type of queries that involves different coverage constraints on the queried attributes. The query latency in such queries is governed by the highest query coverage required, as this would more likely be the one involving the furthest queried node in the data structure. However, because of the distributed nature of the query forwarding and augmentation, the results show that this latency increases linearly as a function of the size of the queried region.

6.2 Mobile Data Indexing

The presented mobility management protocol was implemented on SIDnet-SWANS simulator for WSN (13). The nodes' mobility was assumed under two different mobility models: random and controlled. The controlled mobility refers to a scenario where sensor nodes are



(a) (b)

Figure 28. Number of messages communicated for different accuracies and geometric coverages.

moved based on an underlying application requirement. For our simulations we used the algorithm presented in (26) to compute the coordinates of mobile nodes at each step. In the case of random mobility the new location of each node is computed using a random direction. In addition, we also tested the mobility management protocol under different speeds, ranging from 0.5 m/s to 2 m/s, which is practically used in several WSN systems (41; 33).

For our experiments, we have constructed a K-D tree based hierarchical indexing structure over the sensed field. The index nodes are considered to be static, but would rather be moved according to the borders change, to maintain connectivity with the other nodes in their region. The cycle time in our simulations is 5 seconds, i.e., every 5 seconds, nodes inform their value as well positions to their immediate cluster heads (indexing tree leaf nodes).

We measure the performance of the protocol in terms of following parameters: mobility request

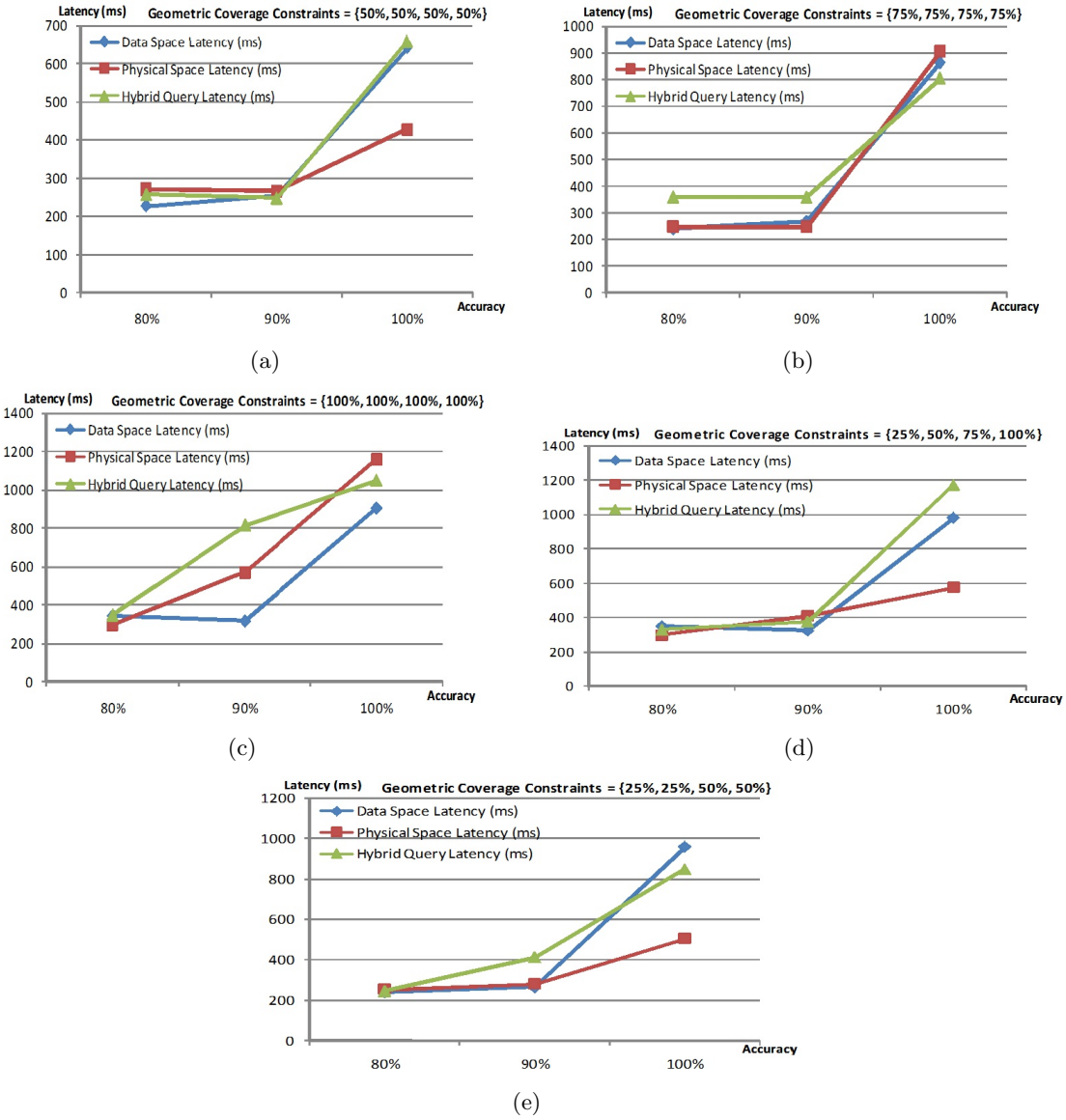


Figure 29. Simulation Results for Queries Containing Different Query Coverage Constraints on Multiple Attributes.

latency, mobility resolution factor, and query latency. Mobility request latency refers to the time it takes for the protocol to adjust the structure to reflect the nodes new positions. Mobility resolution factor (MRF) reflects the percentage of requests that required changes beyond the first level of the indexing hierarchy.

Figure 30 plots the average mobility request latency under different mobility speeds. The performance of both mobility cases is quite stable, where the latency is almost consistent with the change of sensor nodes velocity. The mobility request latency for the controlled mobility scenario (i.e. nodes move towards an events of interest while maintaining coverage (26)) is around 15% higher than the random mobility request latency. This is because the number of mobility request received by the cluster heads in the case of controlled mobility is higher, compared to the random mobility. Note that in the case of random mobility, overall more sensor nodes maybe moving. However, a significant number of consecutive mobility steps may cancel each other, thus keeping the sensor nodes within the same local region. On the other side, in the controlled mobility scenario each sensor node is moving on a specific path towards the target point. Accordingly, with each time step, a node progresses towards moving into or outside of a specific local region, thus requiring mobility adjustment in the indexing structure.

In Figure 31, MRF is shown for different mobility scenarios. The general trend of the MRF is larger for the controlled mobility algorithm, as the nodes following a specific path are able to cause more disturbance in all the regions they pass by, which creates unbalance in multiple local regions. Because of this unbalance, adjustment to mobility may require adjustment at more than one of the hierarchy. The maximum MRF shown for all cases is less than 17%.

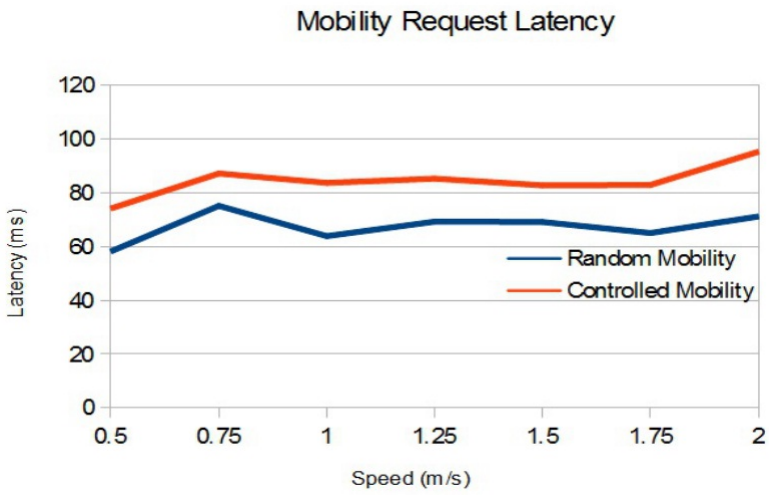


Figure 30. Average latency of incorporating mobile node in the indexing structure Vs. sensor node speed.

Which means that the mobility management protocol is able to resolve successfully over 83% of the mobility requests at the lowest level of the indexing tree, without the need of having this mobility information traverse the whole indexing structure.

Figure 32 compares the latency of different data queries to the mobility managed structure (under random and controlled mobility) and the static structure where the indexing structure does not change itself to accommodate mobility and thus becomes relatively unbalanced. We present results for three different types of queries.

32(a) shows the difference in data-space query latency for static as well mobility manages structures under different mobility scenarios. The static case shows higher costs for achieving more accurate results. This is because on the lower level of the indexing structure, the static scenario

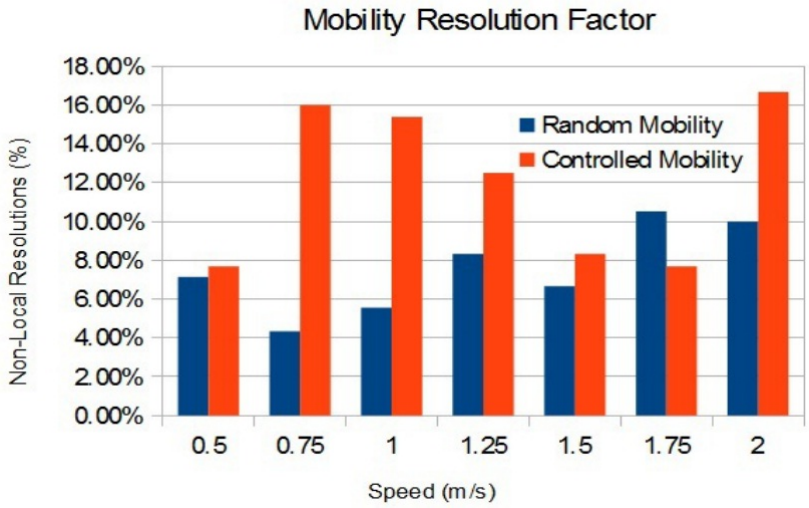


Figure 31. Mobility Resolution Factor (MRF): The percentage of mobility requests that the mobility protocol is unable to resolve at the lowest level of the indexing structure.

would have a higher memory footprint for the congested regions, which requires more processing and communication time. In 32(b), physical space query latency of the static indexing structure almost matches the mobility managed structure under the random mobility scenario for lower accuracy levels, which is slightly higher than the controlled mobility scenario. However for exact queries (i.e., 100% accuracy), which require the indexing structure to get the data from its leaf nodes, static scenario incurs higher query latency costs.

In Figure 32(b), the hybrid query latency can be viewed as a combination of latencies of both physical-space and data-space queries, where it is clear that the incurred latency is higher for the static case when requiring higher accuracy level. These results show the efficiency of appropriately handling mobility, and its effect on query latency for most cases of mobility scenario,

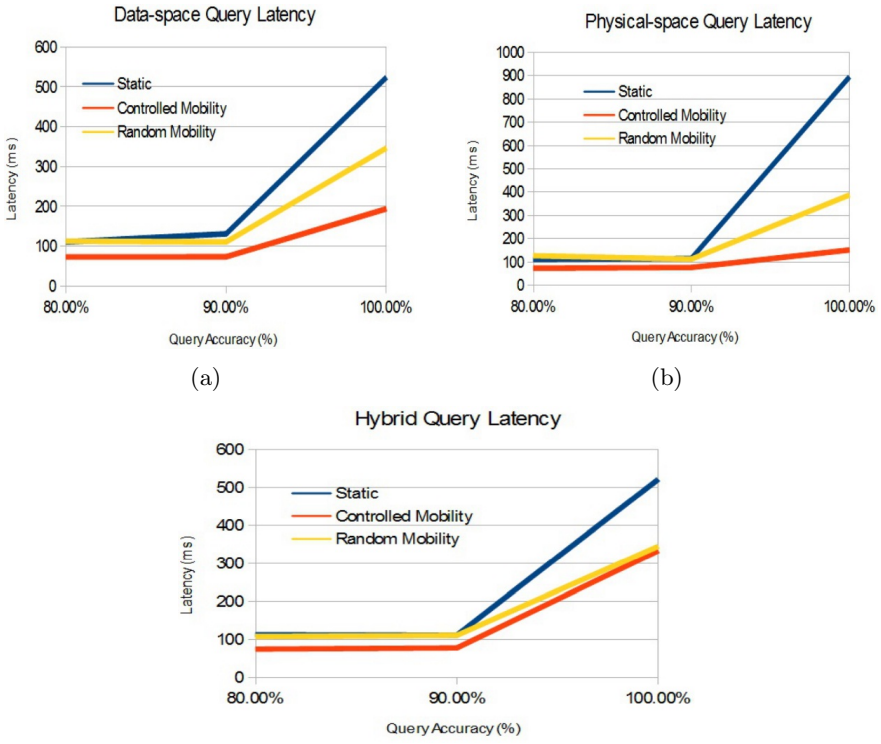


Figure 32. Query latency for (a) data-space, (b) physical-space and (c) hybrid queries Vs. required query response accuracy.

where the static indexing would not be able to provide same latency for queries inquiring higher accuracy, especially for the queries inquiring exact responses.

6.3 Resource Distribution

We now discuss the simulation results illustrating the performance of the presented resource supplying methodologies. The methods were implemented on an orthogonal bisection based K-D Tree implementation (5; 6). The WSN has 500 nodes deployed in a square field of 300x300

meters square, using MAC802.15.4, and Shortest Geographic Path for routing. The power consumption characteristics are based on Mica2 Motes specifications, MPR500CA. The static nodes in the field (S_C) are 80 sensor nodes, within which are the K-D Tree nodes, which leaves the sensed field with 420 mobile nodes (M_Cf). The experiments were run for the various requesting techniques (Centralized, SBDR and SPDR). The number of requested nodes per event was varied from 40 to 200 nodes. The simulated events were up to 12 simultaneous events detected in different clusters. In our experiments, we compare the three requesting methods according to several metrics, which we define as:

- **Request Service Time:** The time elapsed between the issuing of the initial request, till the request is accepted and the nodes are forwarded to the requesting cluster.
- **Average Travel Distance Per Sensor Node:** The average distance each resource (i.e, sensor node) needs to travel across the field to reach the requesting cluster.
- **Communication Cost:** The total number of messages transmitted during the requesting process, including the request messages, response messages and decision messages.
- **Resolution Level:** The leaf-based level in the K-D Tree hierarchy at which the request got accepted. We denote the local-cluster-heads as level 1, global-cluster-heads as level 2, and the sink node as level 3.

In Figure 33, the average request service time for the two distributed methods start lower than the centralized requesting method. However, with the increase in the number of requested resources, their service time exceeds that of the centralized method. This happens because

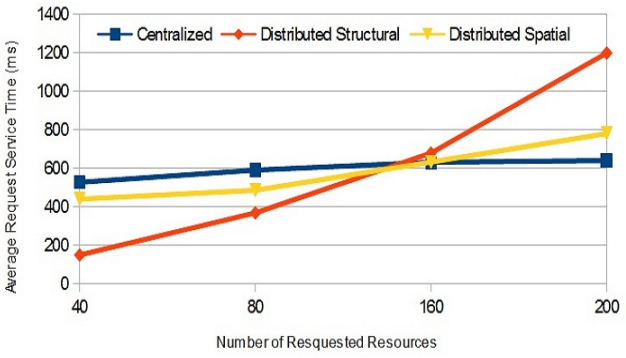


Figure 33. Average Request Service Time against Number of Requested Resources.

the amount of resources available in the clusters neighboring (spatially or structurally) to the requesting cluster cannot suffice this large number of requested resources. Accordingly, further requesting iterations takes place to negotiate resources with more physically distant cluster heads up in the hierarchy. On the other side, the centralized requesting method provides a service time that is mostly stable with the increase of the amount of requested resources.

The average travel distance per sensor node is depicted in Figure 34. The distributed spatial requesting achieves the least average travel distance per sensor node, followed by the distributed structural method. However it seems counter-intuitive that the centralized method does not achieve the most optimal solution, this happens due to the less information it has. The decision in the centralized method is taken at the sink node, which has comprehensive information about the field until the local-cluster-heads level. Thus, it centrally calculates the most optimal distance based on the providing local-cluster-heads locations, which might have the available nodes within their clusters further from the border. On the contrary, the decentralized methods

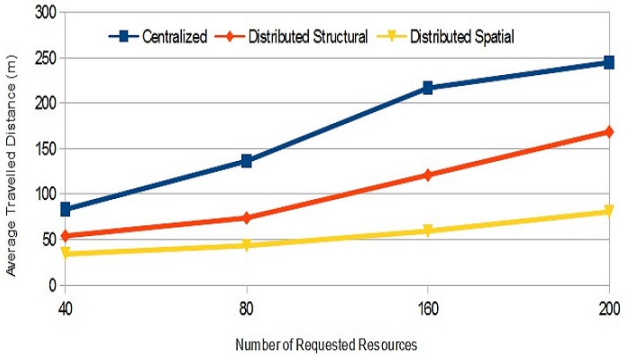


Figure 34. Average Travel Distance Per Sensor Node against Number of Requested Resources.

negotiate the resource supplying process first at the local-cluster-heads level, which makes them able to optimize based on the real location of the sensor nodes rather than the local-cluster-heads locations. If the centralized solution needed to be optimal, which is doable in terms of the logical capability, this would require the requesting information process to include the locations of the potential sensor nodes, which we consider as a significant communication overhead.

The communication cost of the centralized method is higher than the distributed methods, as shown in Figure 35, because of the information gathering rounds. In order to compare the communication cost of the two distributed methods, the resolution level depicted in Figure 36 gives us more clarity about the behavior inside the indexing hierarchy. The distributed methods were able to handle the requests below 160 sensor nodes without the need of propagating the request to the sink node. In this case, the communication cost of the distributed spatial method is higher than the distributed structural method, because of its need to communicate with the

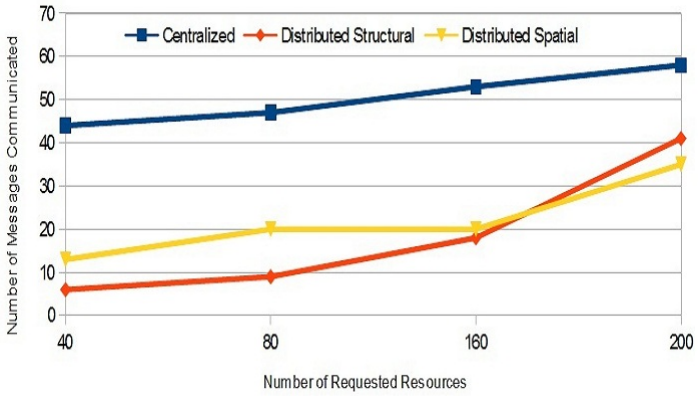


Figure 35. Number of Messages Communicated against Number of Requested Resources.

neighbors list, which is more than the sibling nodes in the K-D Tree. After this threshold, the communication cost of the distributed structural method slightly exceeds the distributed spatial method, because the number of decision messages is more likely to be higher for the structural method. This is because the partial acceptance across the hierarchy eliminates all the neighboring list clusters from being included in any further decision announcements, as they have already sent out their available resources to the requesting cluster.

Figure 37 and Figure 38 compare the three requesting strategies when multiple simultaneous events are detected in different regions of the field, each requesting 30 sensor nodes. Thus, multiple requests are issued to the indexing structures from different leaf nodes (i.e, local-cluster-heads). The centralized method comes to be of the highest request service time and average travel distance, while the other two distributed methods achieve comparable results. The distributed methods outperform the centralized method because of their capability of

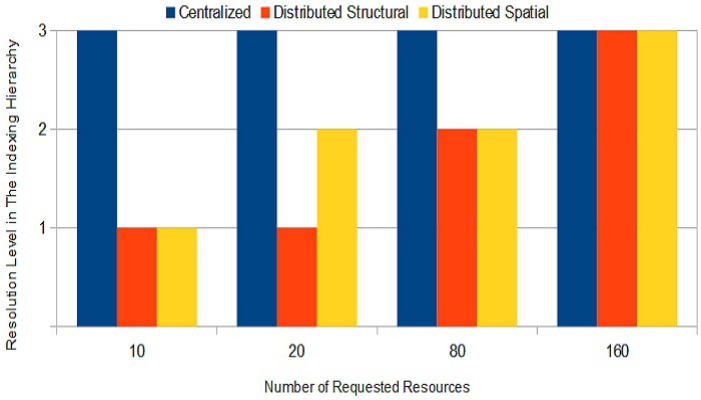


Figure 36. Resolution Level in The Indexing Hierarchy against Number of Requested Resources.

handling the requests within their locality by providing resource supplies from the (spatially or structurally) neighboring clusters.

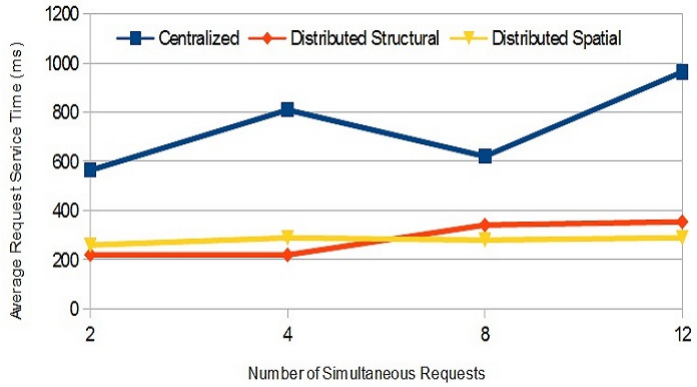


Figure 37. Average Request Service Time against Number of Simultaneous Requests.

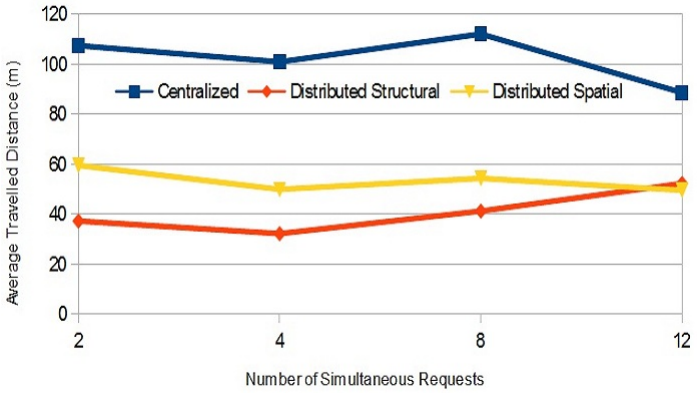


Figure 38. Average Travel Distance Per Sensor Node against Number of Simultaneous Requests.

CHAPTER 7

RELATED WORK

Data indexing in WSN has been studied over the past decade, and several algorithms with different perspectives were presented to solve it. Many of these algorithms did not consider the mobility of sensor nodes. Centralized solutions, as in (42), proposed transmitting data across paths in the network using lifting technique and wavelet based compression. In such methods the network usually suffers from congestion around the sink node, which creates a communication bottleneck, and decreases the lifetime of the nodes in the area around the sink node. Several distributed data indexing algorithms were proposed (43; 44; 45; 15; 46). In (43), a hierarchical data structure is constructed and data is mapped to the indexing structure using geographic hash tables (GHT). This algorithm creates redundancy in data transmission, where the same raw data is reported to multiple nodes in the indexing structure.

In (43; 44), DIMENSIONS, a three level spatio-temporal indexing algorithm is proposed, where it starts by local temporal summarization of sensed values in each node, then locally gathering this data in a grid-based overlay structure. The spatial summaries are then created using Wavelet compression, and forwarded to the sink node. However this multi-layer hierarchical solution provides a good localization that reflects efficient maintenance cost, it lacks representing the data-space information, which makes the system unable to support queries involving data ranges. Also, with increasing data rate, the lossy summarization with no model wont be able to capture the spatial distribution of the sensed data, which results in higher error rates

for representing the sensed information.

Meliou et al. (15) proposed an algorithm with a novel idea for data indexing of sensed values in a hierarchical data structure using approximate modeling. Gaussian models were used in this system to abstract large amount of sensed values and elevate them across the hierarchy, leading to more efficient reporting at the cost of accuracy loss across the hierarchy. Such system lacks the representation of sensor nodes positions, and assumes that Gaussian models are suitable for all types of sensed phenomena, which is not generic enough for a wide range of sensed phenomena not of Gaussian distribution nature. Also, Gaussian models are successful in representing the average behavior of a region, but they lose the information about the extreme (maximum and minimum) sensed values, which are of high interest for many WSN applications.

An approach for constructing approximate spatial summaries and determining boundaries of regions with same sensed values was presented in (47). Although the work addresses the issues of different precision, the results cannot be straightforwardly extended to handle multiple queries with awareness of both physical and data spaces. Another distributed algorithm proposed by Ouksel and Hauswirth (45) indexes the WSN data across a spanning tree according to a key for each node of the spanning tree. Each sensor node identifies its indexing node through a key, which is formed by shuffling the position of the node, along with the values of the different phenomenon being sensed. However this algorithm supports mobility of sensor nodes, it falls short in the maintenance cost of the data updates, as a sensor node may have to update its information at an indexing node that is far from its location. On the other side, if the key is arranged in a way that favors position of sensor node for local region reporting, the sys-

tem doesn't support data-space indexing efficiently. Monitoring the WSN for events have been studied in (48), where an algorithm is proposed to use an optimal number of monitoring nodes and minimize false alarms. Such algorithms are useful for event based monitoring applications, which do not consider aggregating the network data as much as answering specific predicates. In (46), Zhang et al. index-based proposed a data dissemination scheme to address the problem. With this scheme, sensing data are collected, processed and stored at the nodes close to the detecting nodes, and the location information of these storing nodes is pushed to some index nodes, which act as the rendezvous points for sinks and sources. To address the issues of fault tolerance and load balance, the scheme is extended with an adaptive ring-based index (ARI) technique, in which the index nodes for one event type form a ring surrounding the location which is determined by the event type, and the ring can be dynamically reconfigured. In this work, it is presumed that the querying is not going to be for all the data, but rather specific events, which creates an initial analysis phase to decide the existence of events, categorize according to them, and index only the events. This type of algorithms is optimized for some specific applications that are interested in event detection, but not general enough to cover different query forms. It also addresses the issues of false alarms and fault tolerance.

Optimal rate allocation for data aggregation has been studied in (49). In (50), a study for the tradeoff between the number of monitoring nodes and the false alarm rate in the wireless sensor networks is presented. It proposes fully distributed monitoring algorithms, to build up a poller-pollee based architecture with the objective to minimize the number of overall pollers while bounding the false alarm rate. The architecture is build upon the poller-pollee structure,

where sensors self-organize themselves into two tiers, with pollees in the lower tier and pollers in the upper tier. The pollees send status reports to the pollers along multihop paths, during which the intermediate nodes do the aggregation to reduce the message overhead. Each poller makes local decisions based on the received aggregated packets, and forwards its decision towards the sink. Another monitoring algorithm that sends the status reports to different pollers in a round robin manner is presented in (51), where status reports from different pollers are combined to reduce false alarm.

Mobile WSN sink node idea in has taken good consideration in recent research. Controlled mobility have been exploited in several works (52; 53; 54; 55; 56), in which the –one or multiple– sink node(s) moves in the field and gathers the sensed data. Non-hierarchical solutions, as (52; 53; 54; 55), study the optimal path to move across the field, in order to minimize latency. In (56), Xing et al. propose at two tier system of mobile sink node which collects data from static rendezvous points that collect sensed data locally within their vicinity. This clustered data gathering approach increases the efficiency of data gathering and scheduling for sink node mobility, however it doesn't provide a full hierarchical approach for data indexing.

In the recent years, mobility has contributed to the variety of application domains for WSNs and has brought a unique set of challenges and research results (21; 20). From the basic setup-aspect, mobility facilitates deployment (57; 58), augments the monitoring (59; 11) and data gathering (55; 56) capabilities.

But one example of a formalism for relocation of sensor nodes in the deployment phase is using a virtual force based algorithms, proposed in (60; 57; 58). Wang et. at. (58) propose an

iterative algorithm in which coverage holes are detected by sensors using Voronoi diagrams (9). The sensors are then moved from high density zones to low density zones increase coverage. Many sensor relocation algorithms have been proposed (61; 34; 62; 60; 59; 63; 64), presenting distributed algorithms in which the sensor nodes coordinate the relocation process themselves. While these approaches are useful for low density and small scale WSN, in this work we tried to capitalize on a hierarchical structure for settings in which WSNs have larger node population in relatively small spatial regions. The main benefit of our approaches is the separation of the bargaining process (requesting and supplying sensors) from the individual sensor nodes, and elevating is to clusters' level – thus savings in the communication and energy-expenditures.

In (61; 34), Cao et. al. proposed an algorithm, with physical implementation, of a Grid-Quorum solution for sensor relocation in WSN. The sensed field is split into cells arranged in a grid. Each cell has a cluster-head, which known the number of redundant nodes in its area. The information about redundant nodes are shared between cluster heads in the same row and column of the grid. When coverage is required in a specific region, the request is communicated in the row and column of its cell, where the supplier cells are identified using the intersection of the request with the previously advertised redundant nodes. The movement of nodes then follows a cascaded (relayed) path, which is negotiated between the sensor nodes along the path. The single level clustering decreases the scalable performance of the algorithm. If the network size is increased, the advertising and requesting processes will incur high communication cost and latency. The arrangement of cascaded movement in long paths will also be energy intensive, in terms of communication, as it will involve many sensor nodes. Our presented approach, which

operates via hierarchical scheme alleviates some of these drawbacks.

Several fully distributed algorithms were proposed for sensor nodes relocation, for which communication cost would highly increase for large scale and high density WSN. A vector algebra based algorithm to find the locations of potential redundant nodes for coverage compensation is proposed in(62). The selection of the best redundant nodes is performed opportunistically by jointly considering the hole boundaries and the remaining energy of nodes. (60) proposed a distributed algorithm for node deployment and event-based relocation, where sensor nodes are moved by virtual forces. The algorithm requires knowledge of relative positions between neighboring nodes, by which they coordinate their movements. In (59), an iterative distributed relocation algorithm is presented, where each mobile sensor only requires local information in order to optimally relocate itself. The mobile sensors are assumed to be able to move only once over a short distance. Relocation of hopping sensors was investigated in rugged terrains in (63; 64). The mobility model assumes that the sensor nodes move in fixed distance hops, and the algorithms are designed to fill sensing holes by optimizing the required number of hops using direct or relayed motion. Once again, the aspect of our work which complements the contexts addressed in these works is the scalability-benefits.

CHAPTER 8

CONCLUSION

In this dissertation we presented an energy efficient solution for in-network data indexing and querying in WSN. The presented system provides new methods of data and physical space abstractions for data indexing in wireless sensor networks. The physical-space abstraction based on rank order sampling is applicable to a wide range of applications because of its generic nature. The data-space abstraction based on bit maps is first of its kind in indexing in wireless sensor networks.

We proved that the presented system is liberated from organization of data to suit specific query types, or specific data distributions. It rather gives a generic way to answer different queries of various phenomena with high efficiency. It provides a unified information system for multi-attribute sensed fields, which optimizes the in-network storage of sensed data using the presented data abstraction schemes. This facilitates more optimal query processing that is although distributed in its nature, yet capable of being communication efficient.

Fixed size update messages ensure load balancing across the network, reflecting longer network life time. Gathering raw data within geometrically bounded clusters minimizes the communication overhead. Regular sampling of data values gives a suitable generic method for abstraction which provides the network with the capability of estimating sensed values within reasonable error bounds.

In order to show the applicability of the abstraction techniques to various data structures, a

K-D Tree based implementation was implemented as a representative for the BSP data structures. We also presented a hierarchical data indexing and querying system based on Voronoi Treemaps. We showed that the Voronoi Treemaps provide the benefits of a hierarchical data structure that is context aware of the sensed phenomenon, and hence able to provide sensor nodes deployment that is better capable of capturing the phenomenon distribution.

A protocol to manage and maintain in-network indexing structures in WSN under the constraint of mobile nodes was presented. The protocol is applicable to BSP tree structures, where it is based on assigning incrementing values for space splitting borders of the BSP tree. The protocol is based on shrinking and expanding the indexed regions according to the residing number of nodes, in order to keep a balanced load for the indexing structure. The complexity of the presented solution does not exceed a linear order in the size of the indexing structure.

Our results show the capability of handling over 83% of mobility within their local regions of occurrence, without the need of communicating this information across the network. The average latency of balancing the structure in the presence of mobility is in reasonable range. The results also show improvement for query latency results, especially for the higher accuracy queries.

We devised efficient methodologies for scalable management of relocation of mobile sensors in WSNs, in response to a detection of event of interest. The presented work takes into consideration the minimum nodes count needed in each spatial region for guaranteeing certain QoS criteria. Capitalizing on a hierarchical structure, we presented distributed protocols which improve both the response time and the energy consumption due to communication, along with

the choices of nodes to move seeking the optimization of the traveled distance. We presented three different requesting methods (centralized, SBDR and SPDR) and showed the difference in performance between them. The presented approaches are capable of handling simultaneous detection of multiple events. The displacement of the mobile sensor nodes is performed using direct forwarding or relayed motion, which is handled between the cluster heads for large scale management.

CHAPTER 9

FUTURE WORK

In our future work we plan to study a set of problems that pertain to the data indexing and querying problem in WSN. We plan to explore different dimensions as:

- **Dynamic Number of Sensor Nodes:** The assumption of a fixed number of sensor nodes in the field is unrealistic for several WSN applications. Over time, sensors function might stop for a number of reasons, and new sensors are introduced/deployed in the sensed field. Accordingly, the data indexing, mobility management, and resource distributions algorithms need to be adaptable to handle the cases of varying number of sensor nodes in the field, and to be able to achieve the best possible performance, from the perspectives of coverage, monitoring, event handling, ..etc.
- **Resource Distribution:** We plan to investigate what are the costs involved in adjusting different hierarchical structures (e.g., Voronoi Treemaps (7; 8)) when nodes move in response to an event, and develop efficient algorithms for optimizing those costs and identify the trade-offs involved. We also plan to investigate the problem of optimizing the motion plans of the nodes when the budget of available nodes across the network is not sufficient to cater to all the detected events.
- **Communication Cost Optimization:** We plan to investigate the data suppression techniques, and incorporating the implicit clustering algorithms into the indexing struc-

ture. In such way the data indexing system can be more energy efficient, and it would also prolong network lifetime.

CITED LITERATURE

1. Zhao, F. and Guibas, L.: Wireless Sensor Networks: An Information Processing Approach. San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 2004.

2. Dietrich, I. and Dressler, F.: On the lifetime of wireless sensor networks. ACM Trans. Sen. Netw., 5(1):5:1–5:39, February 2009.

3. Mohamed, M. M. A. and Khokhar, A. A.: Dynamic indexing system for spatio-temporal queries in wireless sensor networks. In Mobile Data Management (MDM), 2011 12th IEEE International Conference on, volume 2, pages 35–37. IEEE, 2011.

4. Ali Mohamed, M. M., Khokhar, A., Trajcevski, G., Ansari, R., and Ouksel, A.: Approximate hybrid query processing in wireless sensor networks. In Proceedings of the 20th International Conference on Advances in Geographic Information Systems, pages 542–545. ACM, 2012.

5. Samet, H.: Applications of spatial data structures: Computer graphics, image processing, and GIS. Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 1990.

6. Samet, H.: The design and analysis of spatial data structures. Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 1990.

7. Balzer, M., Deussen, O., and Lewerentz, C.: Voronoi treemaps for the visualization of software metrics. In Proceedings of the 2005 ACM symposium on Software visualization, pages 165–172. ACM, 2005.

8. Balzer, M. and Deussen, O.: Voronoi treemaps. In IEEE Symposium on Information Visualization (InfoVis). IEEE, 2005.

9. De Berg, M., Cheong, O., van Kreveld, M., and Overmars, M.: Computational geometry. Springer, 2008.

10. Mohamed, M. M. A., Khokhar, A. A., and Trajcevski, G.: Voronoi trees for hierarchical in-network data and space abstractions in wireless sensor netowrks. In Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems, pages 207–210. 2013.

11. Mohamed, M. M. A., Khokhar, A., and Trajcevski, G.: Energy efficient in-network data indexing for mobile wireless sensor networks. In Advances in Spatial and Temporal Databases, pages 165–182. Springer, 2013.

12. Mohamed, M. M. A., Khokhar, A., and Trajcevski, G.: Energy efficient resource distribution for mobile wireless sensor networks. In Mobile Data Management (MDM), 2014 IEEE 15th International Conference on, volume 2, pages 49–54. IEEE, 2014.

13. Ghica, O. C., Trajcevski, G., Scheuermann, P., Bischof, Z., and Valtchanov, N.: Sidnet-swans: a simulator and integrated development platform for sensor networks applications. In Proceedings of the 6th ACM conference on Embedded network sensor systems, SenSys '08, pages 385–386, New York, NY, USA, 2008. ACM.

14. http://jist.ece.cornell.edu/index.html.

15. Meliou, A., Guestrin, C., and Hellerstein, J. M.: Approximating sensor network queries using in-network summaries. In Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, IPSN '09, pages 229–240, Washington, DC, USA, 2009. IEEE Computer Society.

16. Chui, C. K.: An introduction to wavelets. San Diego, CA, USA, Academic Press Professional, Inc., 1992.

17. Shi, H. and Schaeffer, J.: Parallel sorting by regular sampling. Journal of Parallel and Distributed Computing, 14(4):361–372, 1992.

18. Gold, C. and Angel, P.: Voronoi hierarchies. In Geographic Information Science, pages 99–111. Springer, 2006.

19. Johnson, B. and Shneiderman, B.: Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In Visualization, 1991. Visualization'91, Proceedings., IEEE Conference on, pages 284–291. IEEE, 1991.

20. Ekici, E., Gu, Y., and Bozdag, D.: Mobility-based communication in wireless sensor networks. Communications Magazine, IEEE, 44(7):56–62, 2006.

21. Pileggi, S. F., Fernandez-Llatas, C., and Meneu, T.: Evaluating mobility impact on wireless sensor network. In Proceedings of the 2011 UKSim 13th International Conference on Modelling and Simulation, UKSIM '11, pages 461–466, Washington, DC, USA, 2011. IEEE Computer Society.

22. Shah, R., Roy, S., Jain, S., and Brunette, W.: Data mules: modeling a three-tier architecture for sparse sensor networks. In Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on, pages 30–41, 2003.

23. Chakrabarti, A., Sabharwal, A., and Aazhang, B.: Using predictable observer mobility for power efficient design of sensor networks. In Proceedings of the 2nd international conference on Information processing in sensor networks, IPSN'03, pages 129–145, Berlin, Heidelberg, 2003. Springer-Verlag.

24. Somasundara, A. A., Ramamoorthy, A., and Srivastava, M. B.: Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In Proceedings of the 25th IEEE International Real-Time Systems Symposium, RTSS '04, pages 296–305, Washington, DC, USA, 2004. IEEE Computer Society.

25. Mulligan, R. and Ammari, H. M.: Coverage in wireless sensor networks: a survey. Network Protocols and Algorithms, 2(2):27–53, 2010.

26. Caicedo-Nuez, C. and Zefran, M.: A coverage algorithm for a class of non-convex regions. In Decision and Control, 2008. CDC 2008. 47th IEEE Conference on, pages 4244–4249, 2008.

27. Inaba, M., Katoh, N., and Imai, H.: Applications of weighted voronoi diagrams and randomization to variance-based k-clustering. In Proceedings of the tenth annual symposium on Computational geometry, pages 332–339. ACM, 1994.

28. Aurenhammer, F. and Edelsbrunner, H.: An optimal algorithm for constructing the weighted voronoi diagram in the plane. Pattern Recognition, 17(2):251–257, 1984.

29. Sud, A., Fisher, D., and Lee, H.-P.: Fast dynamic voronoi treemaps. In Voronoi Diagrams in Science and Engineering (ISVD), 2010 International Symposium on, pages 85–94. IEEE, 2010.

30. Nocaj, A. and Brandes, U.: Computing voronoi treemaps: Faster, simpler, and resolution-independent. In Computer Graphics Forum, volume 31, pages 855–864. Wiley Online Library, 2012.

31. Zhang, W., Cao, G., and Porta, T. L.: Data dissemination with ring-based index for wireless sensor networks. IEEE Trans. Mob. Comput., 6(7):832–847, 2007.

32. Fang, Q., Gao, J., and Guibas, L. J.: Locating and bypassing holes in sensor networks. MONET, 11(2):187–200, 2006.

33. Dantu, K., Rahimi, M., Shah, H., Babel, S., Dhariwal, A., and Sukhatme, G.: Robomote: enabling mobility in sensor networks. In Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on, pages 404–409, 2005.

34. Teng, J., Bolbrock, T., Cao, G., and Porta, T. L.: Sensor relocation with mobile sensors: Design, implementation, and evaluation. In MASS, pages 1–9, 2007.

35. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E.: Wireless sensor networks: a survey. Computer Networks, 38(4):393–422, 2002.

36. Adaikkalavan, R. and Chakravarthy, S.: Formalization and detection of events using interval-based semantics. In COMAD, pages 58–69, 2005.

37. Trajcevski, G., Scheuermann, P., and Brönnimann, H.: Mission-critical management of mobile sensors: or, how to guide a flock of sensors. In DMSN, pages 111–118, 2004.

38. Hosking, J. R. M.: Modeling Persistence In Hydrological Time Series Using Fractional Differencing. Water Resources Research, 20:1898–1908, December 1984.

39. Keshner, M. S.: 1/f noise. Proceedings of the IEEE, 70(3):212–218, 1982.

40. Willinger, W., Taqqu, M., Sherman, R., and Wilson, D.: Self-similarity through high-variability: statistical analysis of ethernet lan traffic at the source level. Networking, IEEE/ACM Transactions on, 5(1):71–86, 1997.

41. Pon, R., Batalin, M., Gordon, J., Kansal, A., Liu, D., Rahimi, M., Shirachi, L., Yu, Y., Hansen, M., Kaiser, W., Srivastava, M., Sukhatme, G., and Estrin, D.: Networked infomechanical systems: a mobile embedded networked sensor platform. In Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on, pages 376–381, 2005.

42. Ciancio, A., Pattem, S., Ortega, A., and Krishnamachari, B.: Energy-efficient data representation and routing for wireless sensor networks based on a distributed wavelet compression algorithm. In Proceedings of the 5th international conference on Information processing in sensor networks, IPSN '06, pages 309–316, New York, NY, USA, 2006. ACM.

43. Greenstein, B., Estrin, D., Govindan, R., Ratnasamy, S., and Shenker, S.: Difs: a distributed index for features in sensor networks. In Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on, pages 163–173, 2003.

44. Ganesan, D., Greenstein, B., Estrin, D., Heidemann, J., and Govindan, R.: Multiresolution storage and search in sensor networks. Trans. Storage, 1(3):277–315, August 2005.

45. Ouksel, A. M. and Hauswirth, M.: Dynamically self-organizing sensors as virtual in-network aggregators and query processors in mobile ad-hoc sensor databases.

46. Zhang, W., Cao, G., and La Porta, T.: Data dissemination with ring-based index for wireless sensor networks. In Network Protocols, 2003. Proceedings. 11th IEEE International Conference on, pages 305–314. IEEE, 2003.

47. Gandhi, S., Hershberger, J., and Suri, S.: Approximate isocontours and spatial summaries for sensor networks. In Proceedings of the 6th international conference on Information processing in sensor networks, IPSN '07, pages 400–409, New York, NY, USA, 2007. ACM.

48. Liu, C. and Cao, G.: Distributed monitoring and aggregation in wireless sensor networks. In INFOCOM, 2010 Proceedings IEEE, pages 1–9, 2010.

49. Su, L., Gao, Y., Yang, Y., and Cao, G.: Towards optimal rate allocation for data aggregation in wireless sensor networks. In Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing, page 19. ACM, 2011.

50. Liu, C. and Cao, G.: Distributed monitoring and aggregation in wireless sensor networks. In INFOCOM, 2010 Proceedings IEEE, pages 1–9. IEEE, 2010.

51. Liu, C. and Cao, G.: A multi-poller based energy-efficient monitoring scheme for wireless sensor networks. In INFOCOM 2009, IEEE, pages 2781–2785. IEEE, 2009.

52. Gandham, S., Dawande, M., Prakash, R., and Venkatesan, S.: Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE, volume 1, pages 377–381 Vol.1, 2003.

53. Luo, J. and Hubaux, J.-P.: Joint mobility and routing for lifetime elongation in wireless sensor networks. In INFOCOM 2005. 24th annual joint conference of the IEEE

computer and communications societies. Proceedings IEEE, volume 3, pages 1735–1746. IEEE, 2005.

54. Wang, Z. M., Basagni, S., Melachrinoudis, E., and Petrioli, C.: Exploiting sink mobility for maximizing sensor networks lifetime. In System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on, pages 287a–287a. IEEE, 2005.

55. Hanoun, S., Creighton, D., and Nahavandi, S.: Decentralized mobility models for data collection in wireless sensor networks. In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pages 1030–1035. IEEE, 2008.

56. Xing, G., Li, M., Wang, T., Jia, W., and Huang, J.: Efficient rendezvous algorithms for mobility-enabled wireless sensor networks. Mobile Computing, IEEE Transactions on, 11(1):47–60, 2012.

57. Heo, N. and Varshney, P. K.: Energy-efficient deployment of intelligent mobile sensor networks. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 35(1):78–92, 2005.

58. Wang, G., Cao, G., and La Porta, T.: Movement-assisted sensor deployment. Mobile Computing, IEEE Transactions on, 5(6):640–652, 2006.

59. Wang, W., Srinivasan, V., and Chua, K.-C.: Coverage in hybrid mobile sensor networks. Mobile Computing, IEEE Transactions on, 7(11):1374–1387, 2008.

60. Garetto, M., Gribaudo, M., Chiasserini, C.-F., and Leonardi, E.: A distributed sensor relocatlon scheme for environmental control. In Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on, pages 1–10. IEEE, 2007.

61. Wang, G., Cao, G., La Porta, T., and Zhang, W.: Sensor relocation in mobile sensor networks. In INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, volume 4, pages 2302–2312. IEEE, 2005.

62. Qin, N.-n., Guo, L.-x., Ding, Z.-g., and Xu, B.-g.: A vector algebraic algorithm for coverage compensation in hybrid wireless sensor networks. International Journal of Distributed Sensor Networks, 2013, 2013.

63. Kim, M., Mutka, M. W., and Choo, H.: On relocation of hopping sensors for rugged terrains. In Computational Science and Its Applications (ICCSA), 2010 International Conference on, pages 203–210. IEEE, 2010.

64. Pei, Y., Cintrón, F. J., Mutka, M. W., Zhao, J., and Xi, N.: Hopping sensor relocation in rugged terrains. In Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, pages 3856–3861. IEEE, 2009.

65. Liu, C. and Cao, G.: Distributed monitoring and aggregation in wireless sensor networks. In INFOCOM, pages 2097–2105, 2010.

66. Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W.: Tinydb: an acquisitional query processing system for sensor networks. ACM Trans. Database Syst., 30(1):122–173, 2005.

APPENDICES

IEEE Publishing Permissions

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you must follow the requirements listed below:

Textual Material: Using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line 2011 IEEE.

In the case of illustrations or tabular material, we require that the copyright line [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.

If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior authors approval.

https://www.ieee.org/publications_standards/publications/rights/permissions_faq.pdf

ACM Publishing Permission

The publishing permission for the paper (10) are found in Figure 39.

This is a License Agreement between Mohamed M.Ali Mohamed ("You") and Association for Computing Machinery, Inc. ("Association for Computing Machinery, Inc.") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Association for Computing Machinery, Inc., and the payment terms and conditions.

License Number	3626050285408
License date	May 11, 2015
Licensed content publisher	Association for Computing Machinery, Inc.
Licensed content publication	Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems
Licensed content title	Voronoi trees for hierarchical in-network data and space abstractions in wireless sensor networks
Licensed content author	Mohamed M. Ali Mohamed, et al
Licensed content date	Nov 3, 2013
Type of Use	Thesis/Dissertation
Requestor type	Author of this ACM article
Is reuse in the author's own new work?	Yes
Format	Electronic
Portion	Excerpt (> 250 words)
Will you be translating?	No
Order reference number	None
Title of your thesis/dissertation	Energy Efficient-In-Network Data Indexing for Wireless Sensor Networks
Expected completion date	May 2015
Estimated size (pages)	120

Figure 39. ACM Publishing Permission for (10).

VITA

NAME	Mohamed Mohamed Ali Mohamed
EDUCATION	B.Sc., Computer Engineering, Cairo University, Cairo, 2007
	M.Sc., Mechanical Engineering, University of Illinois at Chicago, Chicago, Illinois, 2009
	Ph.D., Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, Illinois, 2015
TEACHING	Department of Electrical and Computer Engineering, University of Illinois at Chicago: Lecturer of computer organization undergraduate class, 2014.
	Department of Electrical and Computer Engineering, University of Illinois at Chicago: Teaching assistant for microprocessors, computer organization, digital systems design, and probability theory courses, 2010 – 2013.
	Department of Mechanical and Industrial Engineering, University of Illinois at Chicago: Teaching assistant for introduction to thermodynamics course, 2008.
HONORS	Best Short Paper Award: ACM MSWiM conference 2013, for the paper titled "Voronoi trees for hierarchical in-network data and space abstractions in wireless sensor netowrks".
	AAAEA scholarship award, 2012 – 2014.
	Cairo University academic achievement award, 2003 – 2007.
EXPERIENCE	ASML US Inc. – Platform PEG Engineer: Part of the platform product engineering group which interface with customers R&D, QA, and Marketing teams to support, enhance and develop lithography software distribution product features, 2015 – Current.

Department of Electrical and Computer Engineering, University of Illinois at Chicago – Research Assistant: Develop energy efficient distributed indexing algorithms and query processing for static and mobile wireless sensor networks, 2010 – 2014.

LitePoint Corporation – Software Testing Summer Intern: Contributed in test automation of cellular communication solution for verification device (GSM/3G/LTE), Summer 2012, 2013.

Department of Mechanical and Industrial Engineering, University of Illinois at Chicago – Research Assistant: Worked in developing multi-body dynamics simulation software, and finite elements analysis, 2009.

Mentor Graphics Corporation – Software Development Engineer: Worked in Litho-Friendly Design tool to manage lithographic process variability in the early stages of design creation. Collaborated with the technical marketing team to create new product features based on marketing analysis, 2007 – 2008.

PUBLICATIONS

Mohamed, M. M. A., Khokhar, A., and Trajcevski, G.: Energy efficient in-network data indexing and query processing for static wireless sensor netowrks. Journal, Accepted for publication in International Journal of Next-Generation Computing (IJNGC) 2015.

Mohamed, M. M. A., Khokhar, A., and Trajcevski, G.: Energy efficient resource distribution for mobile wireless sensor netowrks. In Mobile Data Management (MDM), 2014 IEEE 15th International Conference on, volume 2, pages 49–54. IEEE, 2014

Mohamed, M. M. A., Khokhar, A., and Trajcevski, G.: Energy efficient in-network data indexing for mobile wireless sensor netowrks. In Advances in Spatial and Temporal Databases, pages 165–182. Springer, 2013

Mohamed, M. M. A., Khokhar, A. A., and Trajcevski, G.: Voronoi trees for hierarchical in-network data and space abstractions in wireless sensor netowrks. In Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems, pages 207–210. 2013.

Ali Mohamed, M. M., Khokhar, A., Trajcevski, G., Ansari, R., and Ouksel, A.: Approximate hybrid query processing in wireless sensor networks. In Proceedings of the 20th International Conference on Advances in Geographic Information Systems, pages 542–545. ACM, 2012.

Mohamed, M. M. A. and Khokhar, A. A.: Dynamic indexing system for spatio-temporal queries in wireless sensor networks. In Mobile Data Management (MDM), 2011 12th IEEE International Conference on, volume 2, pages 35–37. IEEE, 2011.