Energy Efficient Data Indexing and Query Processing for Static and Mobile Wireless Sensor Networks

MOHAMED M.ALI MOHAMED, ASHFAQ KHOKAR, GOCE TRAJCEVSKI
University of Illinois at Chicago, Northwerstern University

This work addresses the problem of efficiently balancing the use of network resources when processing both spatially-constrained and (sensed) value based queries in Wireless Sensor Networks. To alleviate the drawbacks inherent to centralized approaches e.g., overheads in energy consumption and latency due to the transmission of the individual raw data/measurements to a dedicated sink, we propose in-network processing methodologies which unify the management of physical and data-space based queries. Since sensed data typically represents values that evolve over time, the distributed data management approaches need to be efficient in terms of communication cost and storage requirements. Furthermore, if the query processing paradigm(s) allows approximate answers, it may yield additional benefits if data abstractions based on higher-order statistics are integrated in the data management. The related challenges are further compounded if the nodes are mobile, for the purpose of adapting the quality of sensing/coverage to spatial changes in the data field. We present novel communication and storage efficient physical and data-space abstractions to facilitate in-network indexing of sensed data and processing of queries in WSNs consisting of mobile and static nodes. We also present novel algorithms to handle changes in the abstractions due to mobility of the nodes. To trade-off (im)precision vs. energy consumption, the proposed abstraction schemes combine rank order statistics, regular sampling, and bitmap representation. The proposed abstractions are generic, in the sense that they can be utilized in any hierarchical indexing structure that is based on binary space partitioning (BSP), such as k-d trees, Quadtrees and Octrees. Based on implementation in SIDnet-SWANS simulator, our experimental results demonstrate the effectiveness of the proposed abstractions under different mobility models, mobility speeds, and query streams.

Keywords: Distributed Algorithms, Mobility, Wireless Sensor Networks, Data Indexing, Query Processing

1. INTRODUCTION

In the recent years, Wireless Sensor Networks (WSNs) have become a technology of choice in an increasing number of application domains, spanning from environmental monitoring, health-care, structural safety assurances and military battlefields [Zhao et al. 2004]. WSNs provide the capability of monitoring and reporting up to date information for various physical phenomena, and even provide the capabilities to perform actuation in response to detected values [Chen et al. 2011] (and the references therein). Regardless of the application contexts, efficient use of the battery power of sensor nodes (also referred to as motes) in WSNs remains one of the major concerns [Zhao et al. 2004]. As WSNs may often operate in harsh environmental conditions and/or inaccessible locations, supplying power to the motes after deployment becomes difficult. Thus, smart exploitation of the energy reserves, which in many respects directly implies prolonging the networks operational lifetime, is a challenge for almost every WSN application.
In addition to the basic capability of monitoring a phenomenon of interest, WSNs are also charged with the task of responding to queries from network users. A brute force approach to query processing is a centralized approach where the sensed values are transmitted to central node (referred to as sink node), and then the sink node responds to all the queries. This approach has its inherent limitations and drawbacks, including latency, single point of failure, and bottlenecks. WSN can also operate by creating aggregated in-network organization and storage of sensed information. Data indexing in WSN aims at creating an in-network communication and storage methodolo-

This research was supported in part by the National Science Foundation grants CNS 0910988, 0910952, III 1213038, and ONR N00014-14-10215.

gies, which assist the objective of monitoring and responding to dispatched queries, hopefully with smaller energy overheads [Akkaya et al. 2005]. Since the communication between nodes consumes 2-3 orders of magnitude more energy than sampling/sensing and calculations, the crucial aspect becomes sharing as much useful information with the least possible communication. Given the complexity of the data indexing problem, existing methodologies [Ciancio et al. 2006; Greenstein et al. 2003; Ganesan et al. 2005; Ouksel et al. 2007; Meliou et al. 2009; Liu et al. 2010] can be categorized along several dimensions. Some of them follow a centralized aggregation method, where all information are pulled towards the sink node [Ciancio et al. 2006]; which is the central node connected to the base station. The centralized approach entails a load balancing problem, where the nodes closer to sink node experience higher communication traffic. Other approaches [Greenstein et al. 2003; Ganesan et al. 2005; Ouksel et al. 2007; Meliou et al. 2009; Liu et al. 2010] aim at creating in-network (logical) data structure among the sensor nodes. In the distributed approach, each sensor node is logically connected to one –or more– node(s) of that data structure, and accordingly reports its sensed information to this node(s). Some data indexing algorithms perform further aggregation of the gathered information, thereby creating hierarchical representation of the field information. When it comes to the particular reported information, it can be categorized along the spectrum of values of the monitored phenomenon, as well as the collection of possible locations at which the values were sensed. Each of these categories, in turn, has an impact on the efficient management of the network resources.

The main motivation for this work is based on the observation that the existing works on data indexing are not capable of creating a system that combines the aggregation of both the the sensed values and their respective locations in an integrated manner. The aggregation techniques applied are either trading-off between the precision sensed values and their respective locations, or designed to better suit a specific data distribution of the sensed phenomenon. Moreover, the existing data indexing approaches do not take mobility of sensor nodes into consideration, although in many applications, WSN have (portion of) the nodes capable to change their locations in the field, to adapt the coverage to the dynamic behavior of the sensed phenomena, or to track objects in the sensed field. In such settings, if the indexing algorithm were not to adapt to mobility, the network might incur significant increase in energy consumption.

In this paper we propose novel generic data abstraction techniques for both sensed values and sensor nodes locations to facilitate in-network indexing of sensed data and processing of queries in WSNs. The proposed techniques handle 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 proposed abstraction techniques are independent from the underlying phenomenon distribution, which renders them to be applicable to various data distributions while preserving load balancing aspect. We also present an energy efficient query traversal algorithm, that is capable of analyzing queries, directing them to the appropriate indexing nodes, and efficiently aggregating their results. The proposed system is generic enough to fit a wide variety of the commonly used spatial data structures. We show how it can 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 data structures based on binary space partitioning (BSP). Our experimental results show the efficiency of the proposed algorithm, in terms of query latency, and maintenance cost.

Earlier versions of this work were published in [Mohamed et al. 2011; Mohamed et al. 2012; Mohamed et al. 2013]. This article extends the previous results in the following aspects: (a) Optimized data-space abstraction for static networks; (d) Extended mobility experimental results; (c) More elaborate abstraction techniques analysis; (d) Query traversal algorithms.

The rest of this article is organized as follows: In Section 2, we discuss the preliminary assumptions and analyze the different aspects of the problem. We follow with presenting the proposed data abstraction techniques in Section 3. Section 4 discusses the details of the proposed query

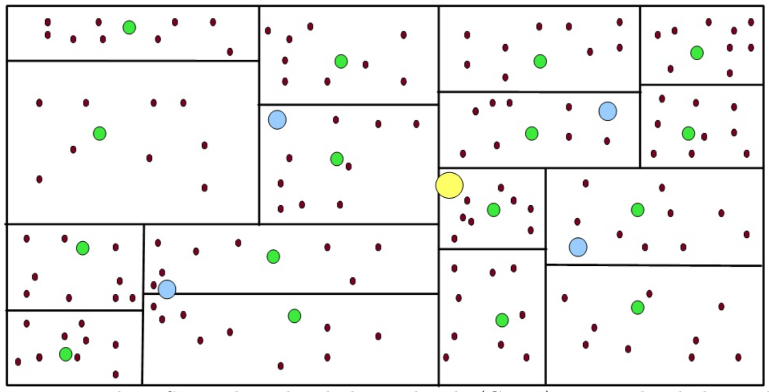


Figure. 1: An OBT 160 node WSN with 16 local cluster heads (Green), 4 next level cluster heads (Blue), and one yellow sink node.

processing methodology. In Section 5 we address the efficient management of the mobility in WSN with combined mobile and static nodes. Experimental results are presented in Section 6. Section 7 discusses the related works and Section 8 concludes the paper and outlines the directions for future work.

2. PRELIMINARIES

In this section we describe the general problem settings and the main aspects of the proposed methodologies. We illustrate the required features in many data structures to which the proposed abstraction methods are applicable . We then present the details of the notation used to describe the queries to the indexing system followed by discussion of the different dimensions of the data indexing problem, and the metrics to be used for the assessment of the proposed system.

2.1 Data Structure Assumptions

The proposed 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 [Samet 1990]. 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 respective indexing data structure, each leaf node knows the borders of the spatial region it covers. Additionally, each leaf node is considered responsible "managing" the sensor nodes within its region. We will be using the name local cluster head interchangeably with the indexing leaf node throughout this paper. Each sensor node will be logically connected to the local cluster head (i.e, indexing leaf node). The local cluster heads are responsible for gathering information from the nodes in their cluster, and applying the first phase of the proposed data abstraction methodology.

2.2 Query Types

WSN queries may inquire values of the sensed phenomena, either in the whole field or in a specific region. In addition, they may also inquire the location(s) from which a value, or a range of values, were reported. Also, the reported values of sensor nodes are generally not accurate due to imperfections and other physical aspects of the sensor nodes. Therefore, approximate query processing paradigm is well-suited to WSN, where a 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:
Q(P,T, C, A), where:

—$\boldsymbol{P}$ denotes the sensed phenomenon (e.g., Temperature, Humidity)

—$\boldsymbol{T}$ denotes a type. We denote the sensed values with $\boldsymbol{T} = \boldsymbol{v}$ and locations type with $\boldsymbol{T} = \boldsymbol{l}$.

—$\boldsymbol{C}$ denotes the type of search bounds for the query: geometric bounds within the sensed field ***(G)***, and/or, either value range within the sensed values ***(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 paper to evaluate previous contributions in solving the indexing problem as well as the proposed solution.
Metric 1: WSN Information Representation
The information in a WSN 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 [Meliou et al. 2009].

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.

3. DATA INDEXING AND QUERY PROCESSING

We now present the details of the proposed abstractions and their use for efficient query processing – along with the protocol specifying the nodes' behavior upon 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 [Dietrich et al. 2009]. 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.

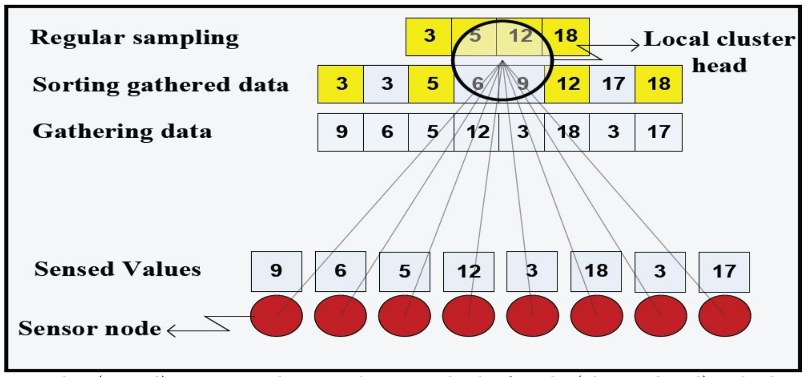


Figure. 2: Sensor nodes (in red) transmit their readings to the leaf node (cluster-head), which sorts the received values and creates the representation construct by regular sampling.

3.1 Physical-space Abstraction

When it comes to physical-space abstractions, the main objective is to provide a hierarchical multi-resolution scheme enabling the generation of approximate answers to 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 $T(V', E')$ of depth d, and *(w.l.o.g.)* a fixed fan-out f, in a graph $G(V, E)$, $E' \subseteq E$, $V' \subseteq V$, where $\forall i$:

—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 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.

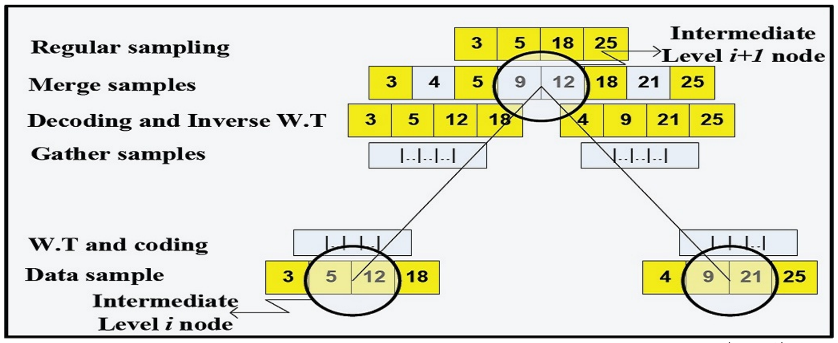


Figure. 3: Physical-space merging process in an intermediate level of the spanning tree ($f = 2$). Lower level nodes compress their samples and transmit them to their parent which decodes, merges them and samples again with rate $1/f$.

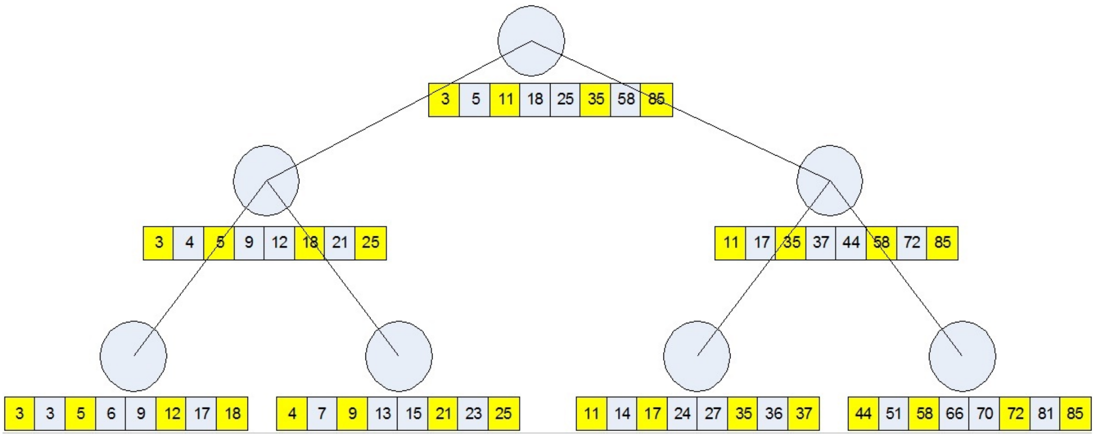


Figure. 4: Physical-space abstraction across multiple levels of a hierarchy with $f = 2$. Each node keeps a sorted array of sensed values within its region and sends a sample of it (yellow background) to its parent, which merges all the samples from its children.

3.1.2 *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 $\nu'_{d,i}$ can apply wavelet transformation [Chui et al. 1992] to its representation construct $p_{\nu'_{d,i}}$, 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 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.

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:

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

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

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

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 [Shi et al. 1992]. 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

$$lb = (i-2)\frac{fD}{k} \tag{3}$$

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

$$ub = (k-i)\frac{fD}{k} \tag{4}$$

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

$$N^{d-1}_{Distributed} = fD - lb - ub = fD - \frac{fD}{k}(k-2) = 2\frac{fD}{k} \tag{5}$$

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

$$E^{d-1}_{Distributed} = \frac{fD}{k} \tag{6}$$

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

$$lb = ((i-2)\frac{fk}{k})\frac{f^{d-m}D}{k} = (i-2)\frac{f^{d-m+1}D}{k} \tag{7}$$

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

$$ub = ((k-i)\frac{fk}{k})\frac{f^{d-m}D}{k} = (k-i)\frac{f^{d-m+1}D}{k} \tag{8}$$

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

$$N^{m-1}_{\mathtt{Distributed}} = f^{d-m+1}D - \frac{f^{d-m+1}D}{k}(k-2) = 2\frac{f^{d-m+1}D}{k} \tag{9}$$

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

$$E^{m-1}_{\mathtt{Distributed}} = \frac{f^{d-m+1}D}{k} \tag{10}$$

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

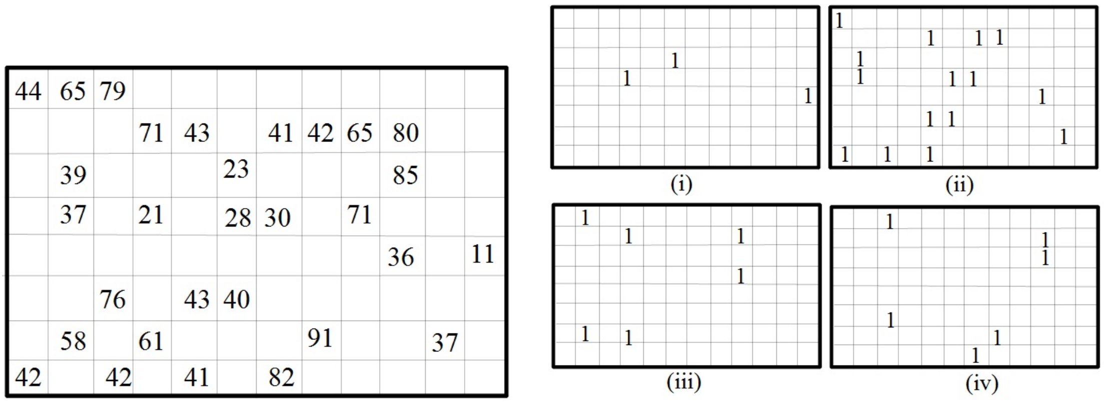
$$N^{j}_{\mathtt{Distributed}} = 2\frac{f^{d-j+1}D}{k} \tag{11}$$

And the error at such level is given by

$$E^{j}_{\mathtt{Distributed}} = \frac{f^{d-j+1}D}{k} \tag{12}$$

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.

3.1.4 *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 $q \geqslant f$. 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



(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.

Figue. 5: Illustrative example of the bitmap creation.

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:

$$RG_l = \{RG_{l0}, RG_{l1}, ..., RG_{lf}\} \tag{13}$$

$$RG_l = \{[min, V_1], [V_1 + \varepsilon, V_2], [V_2 + \varepsilon, V_3], ..., [V_f + \varepsilon, max]\} \tag{14}$$

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.

3.1.5 *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 $L_{\nu'_{d,i}}$ of nodes positions for each leaf node $\nu'_{d,i}$, 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.

3.1.6 *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_l}, the corresponding responsible nodes (f in total) at level i compress

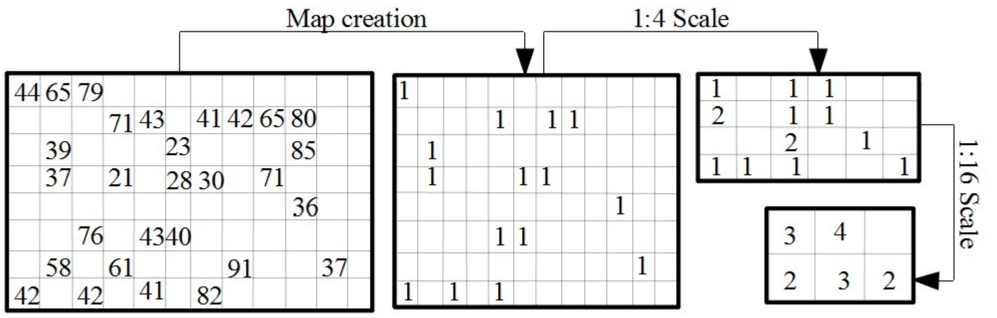


Figure. 6: Illustration example for map creation for data range [26-50], and zooming out twice with 1:4 factors.

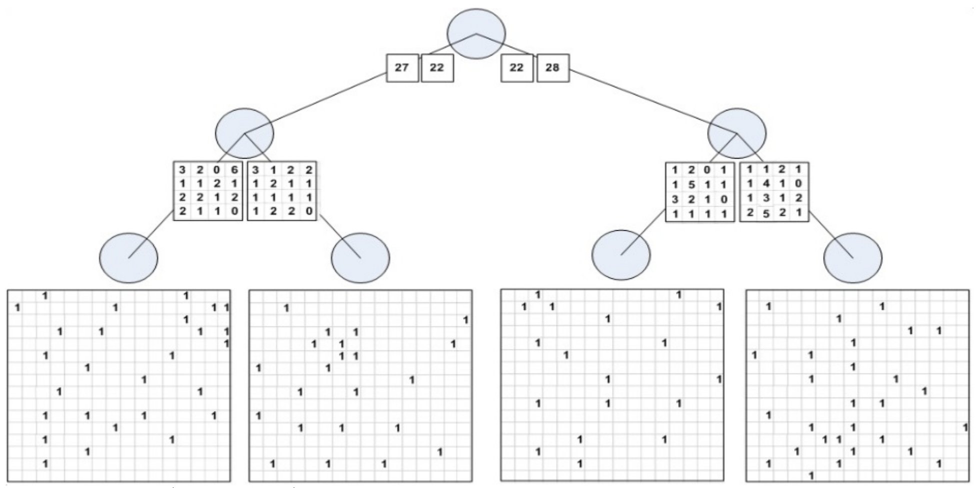


Figure. 7: Data-space (d = 2, f = 2) is split into two ranges. Leaf nodes create maps for their data ranges, and transmit them to the upper level nodes responsible for the same data range. Upper level nodes zoom-out the maps and recursively apply the same process till sink node.

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.

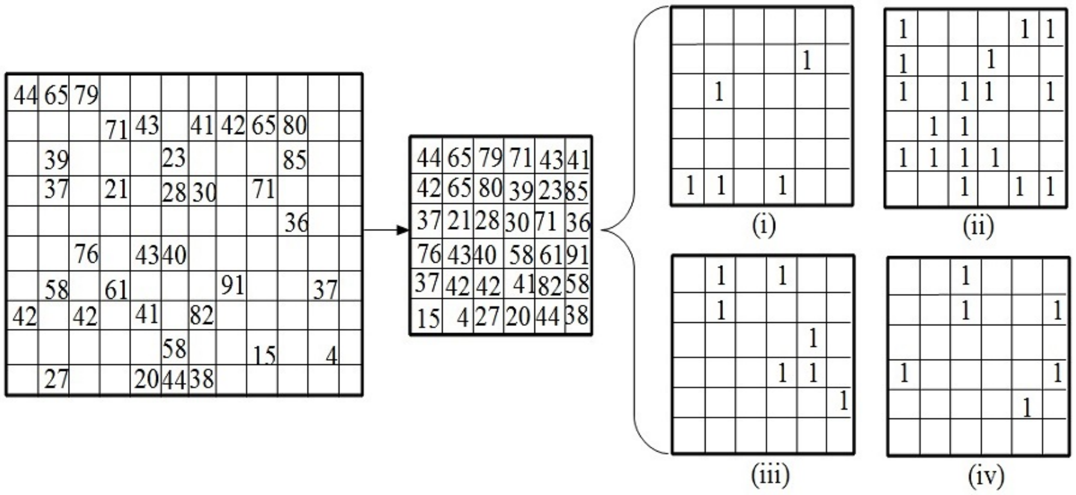


Figure. 8: Illustration example for condensed field representation and its corresponding maps for four data ranges [1-25], [26-50], [51-75], [76-100] respectively.

3.1.7 *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 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.

4. 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.

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 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 1.

Algorithm 1 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 cov-
            erage) 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;
```

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 data-space query responses. The query results are obtained by concatenation of the response maps, while physical-space range queries merge the arrays of sensed values. In the case

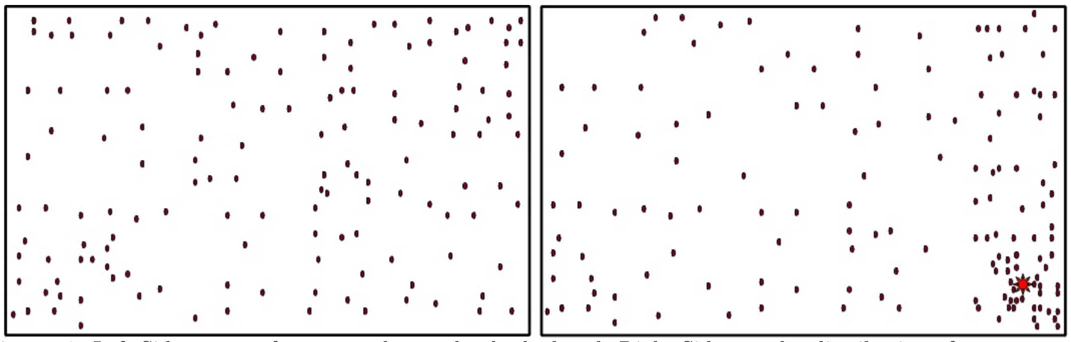


Figure. 9: Left Side - a set of sensor nodes randomly deployed. Right Side – nodes distribution after occurrence of an event of interest in the southeast corner.

of extreme values (min or max) the merging trivially preserves the smallest or largest values. See Algorithm 1 - Part 2 for additional details.

5. MOBILITY MANAGEMENT

5.1 Introduction

Mobile sensor nodes [Ekici et al. 2006; Pileggi et al. 2011] greatly increase the adaptability of the WSNs from different perspectives: (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; (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 [Shah et al. 2003], random [Chakrabarti et al. 2003], or controlled [Somasundara et al. 2004]. For example, in the data coverage problem in WSN [Mulligan et al. 2010], 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 9. The left side of figure 9, 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 [Caicedo-Nuez et al. 2008]) 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 a balanced way.
The main question addressed in this work 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 minimized. 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 [Caicedo-Nuez et al. 2008] as the dictating application for mobility.

5.2 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 assigns an integer rank for each border/hyperplane corresponding to a node in the indexing BSP tree, equal

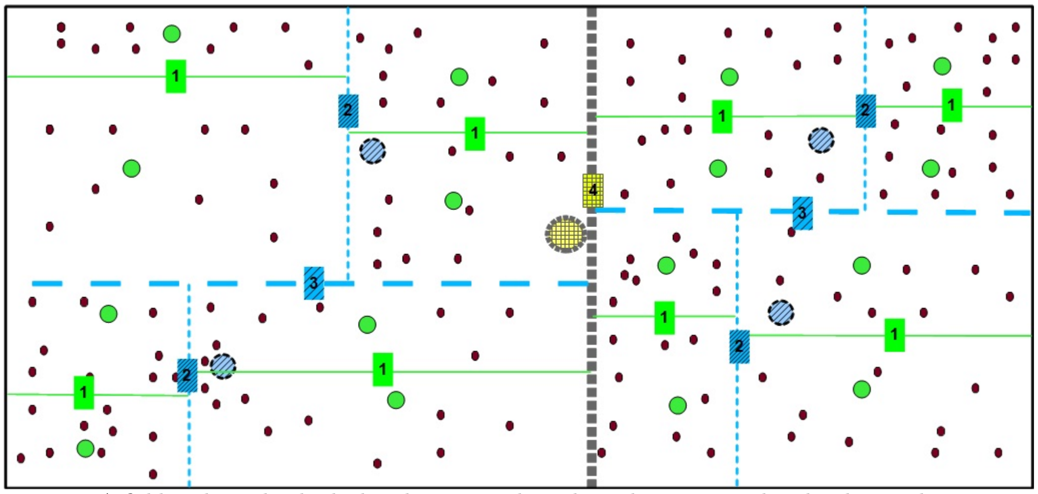


Figure. 10: A field with randomly deployed sensor nodes, where the corresponding borders rank are assigned.

to the depth of the node in the tree (i.e., its level-distance from the root). Figure 10 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 11, which shows a field containing $n = 103$ randomly deployed (small size/red color) sensor nodes. The indexing structure is based on orthogonal bisection [Samet 1990], 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.

5.3 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 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

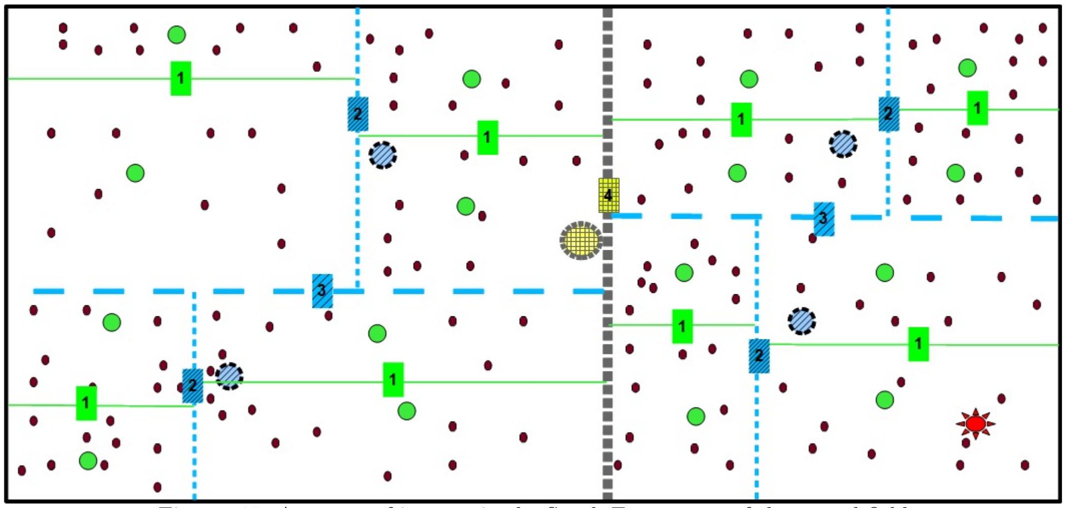


Figure. 11: An event of interest in the South-East corner of the sensed field.

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 2. Algorithm 3 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.

5.4 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 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

Algorithm 2 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)

```
if Sender.depth == Receiver.depth then
  extraNodesCount = Sender.nodesCount - Receiver.optimalNodesCountForCluster;
  if Receiver.nodesCount + extraNodesCount <Receiver.maximumNodesCountForCluster then
    newBorderLocation = calculateNewBorderLocation(Sender.Rank, extraNodesCount);
    send border_change_response(Sender, accepted, Rank, newBorderLocation);
    apply border_change_inform(this, Rank, newBorderLocation);
  else
    send border_change_response(Sender, rejected, Rank, Receiver.nodeCount);
  end if
else
  Receiver.UpdateNodesCount(Sender, Sender.nodesCount);
  if Receiver.nodeCount <Receiver.maximumNodesCountForCluster then
    newBorderLocation = calculateNewBorderLocation(Sender.Rank);
    send border_change_response(Sender, accepted, Rank, newBorderLocation);
    for all childNodes other than Sender do
      send border_change_inform(childNode, Rank, newBorderLocation);
    end for
  else
    Rank = Sender.Rank + 1;
    send border_change_request (Sibling, Rank, Receiver.nodesCount);
    requestingBorderChange = TRUE;
  end if
end if
```

Algorithm 3 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)

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

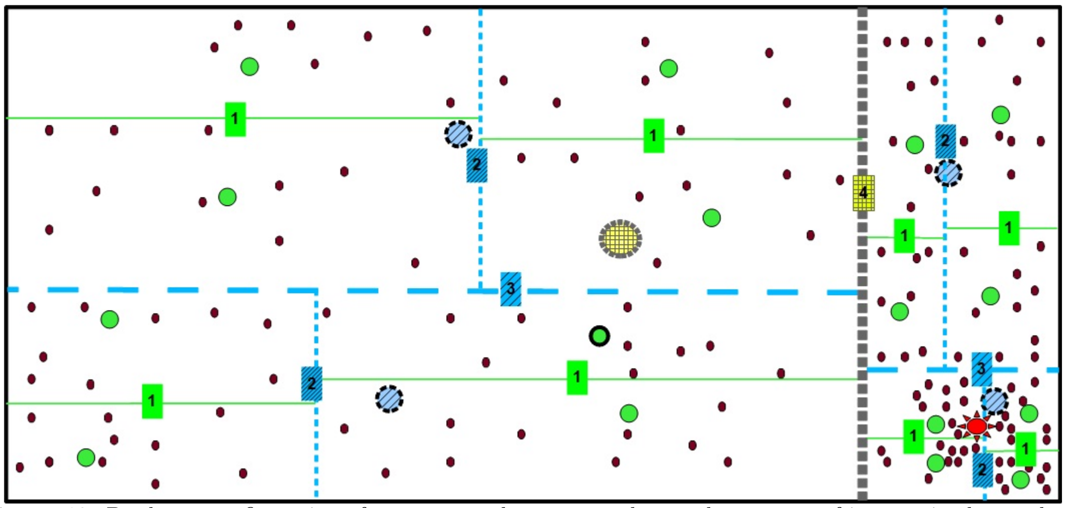


Figure. 12: Borders reconfiguration after sensor nodes are moved towards an event of interest in the southeast corner of the field.

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 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 12 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 4.

Algorithm 4 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
```

Complexity: Algorithm 4 executes when Algorithms 2 and 3 have terminated, and is applied to

all the children of the subtree rooted at the node at which Algorithm 3 has terminated. In the worst-case scenario, the execution of Algorthms 2 and 3, will cause the request to be forwarded 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 4 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.

5.5 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, 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 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.

6. EXPERIMENTAL RESULTS

The proposed abstraction system was simulated using the SIDnet-SWANS WSN simulator [Ghica et al. 2008] based on Jist-SWANS discrete event simulation engine [Jist 2004], 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.

6.1 Indexing Structure Implementation

The presented abstractions methods are generic enough to suit a wide range of data structures. In order to show its applicability and experimental results, we have implemented the proposed abstraction system as an overlay on a well known hierarchical indexing structure K-D tree [Samet et al. 1990]. A detailed discussion of the performance of different data structures with these abstraction methods can be considered for future work.

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 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, $f \times d$ 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 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 13 depicts the communication in data-space abstraction for the first data range of the data-space.

6.2 Simulation Results

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. [Meliou et al. 2009]. 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. 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)***).

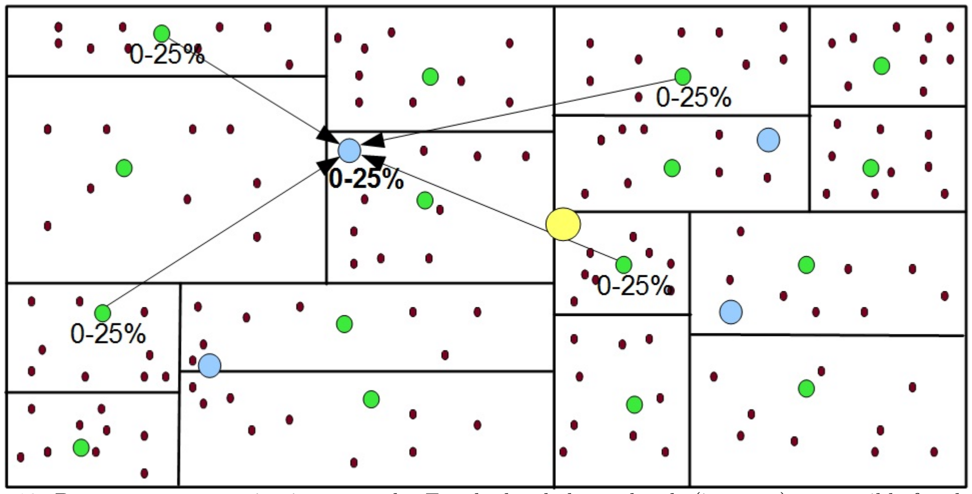


Figure. 13: Data-space communication example. For the local cluster heads (in green) responsible for the lowest data range (i.e: 0-25%), maps are transmitted to the upper level cluster head (in blue) responsible for the same data range.

(2) Data-space queries – asking about the position of nodes sensing some specific data range (i.e: $\boldsymbol{Q(l,\ R,\ A)}$).

(3) Hybrid queries – merging the first two types together by creating a physical space query over a bounded data range (i.e: $\boldsymbol{Q(v,\ G,\ R,\ A)}$ or $\boldsymbol{Q(l,\ G,\ R,\ A)}$).

The approximation error calculated for the proposed method, and the Gaussian approximation method by [Meliou et al. 2009] is based on normalized root mean square error (NRMSE), In the case of [Meliou et al. 2009] 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 14, 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 [Meliou et al. 2009]. Nonetheless, because the data in different regions of the fields may follow different distribution functions (cf. [Hosking et al. 1984; Keshner et al. 1982; Willinger et al. 1997]), the method used in [Meliou et al. 2009] fails to capture the data with the same efficiency as our presented method.

The results of the other distributions in figure 14 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 15 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.

6.2.1 *Proposed Method Results.* 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, and data ranges.

The single-attribute-query results, depicted in figure 16 and figure 17, show the change in the latency of query response with the change of desired accuracy, and geometric bounds (represented

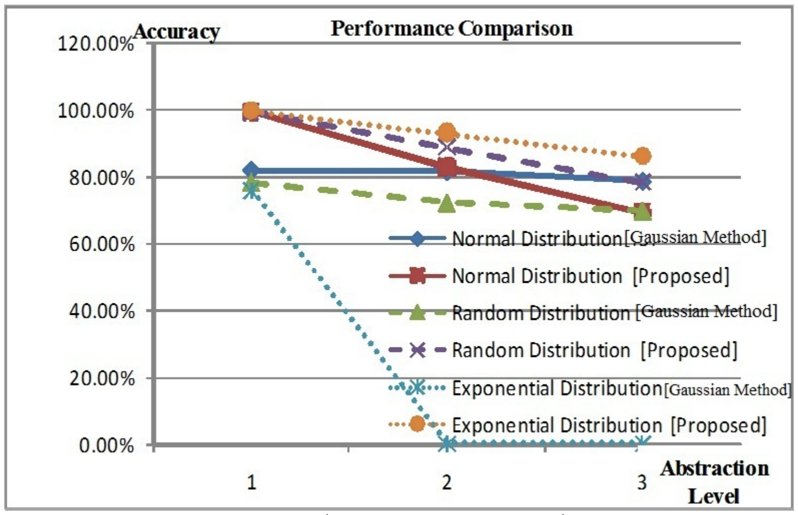


Figure. 14: Approximation error comparison (Gaussian vs. sampling) for normal, uniform, and exponential distributions.

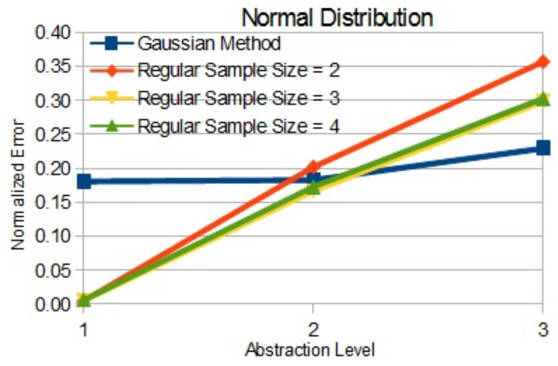
as percentile of the field size). In figure 18, the communication cost is shown for the cases in figure 16 and figure 17.

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.

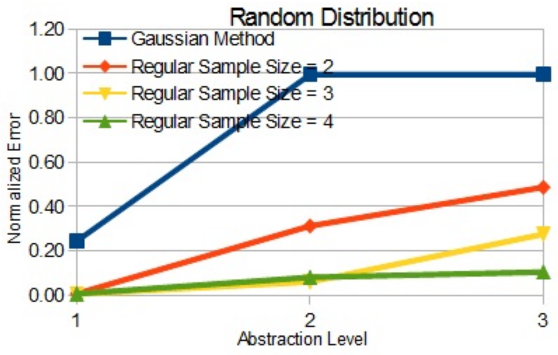
The query latency results for the three types of queries in figure 16 and figure 17 show that the data-space queries have higher latency than the physical space queries when there is no full query coverage. This is due to the nature of locality of the indexing structure nodes for the physical-space abstraction 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 19, 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.

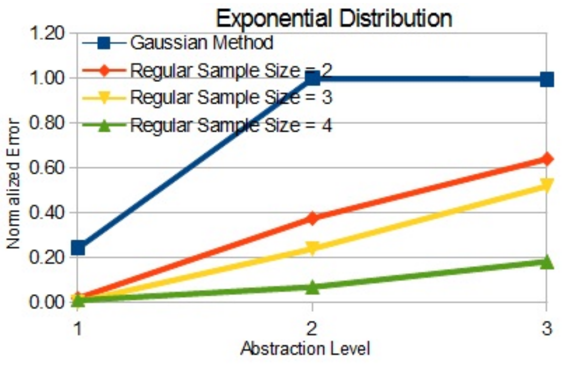
The second part of figure 19 ((d) and (e)) depicts the results for the type of queries that involves different coverage constraints on the queried attributes. The query latency in such queries is governed by the highest query coverage required, as this would more likely be the one involving the furthest 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.



(a) Normal Distribution



(b) Random Distribution

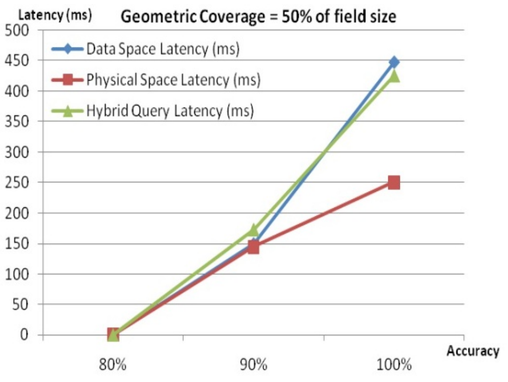


(c) Exponential Distribution

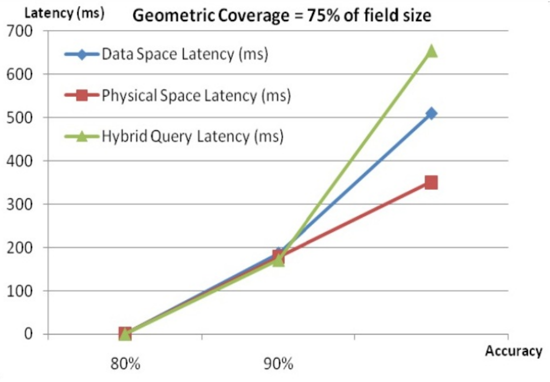
Figure. 15: Approximation error comparison (Gaussian vs. sampling) for different distributions with varying the sample size.

6.3 Mobility Results

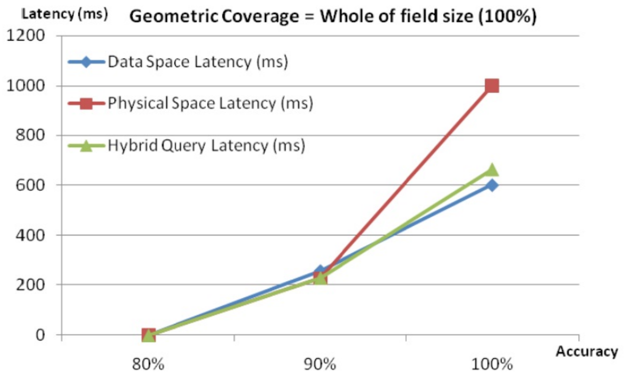
The proposed mobility management protocol was implemented on SIDnet-SWANS simulator for WSN [Ghica et al. 2008]. The nodes' mobility was assumed under two different mobility models: random and controlled. The controlled mobility refers to a scenario where sensor nodes are moved based on an underlying application requirement. For our simulations we used the algorithm presented in [Caicedo-Nuez et al. 2008] 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 [Pon et al. 2005; Dantu et al. 2005]. In our future work we plan to simulate higher speed nodes as



(a)

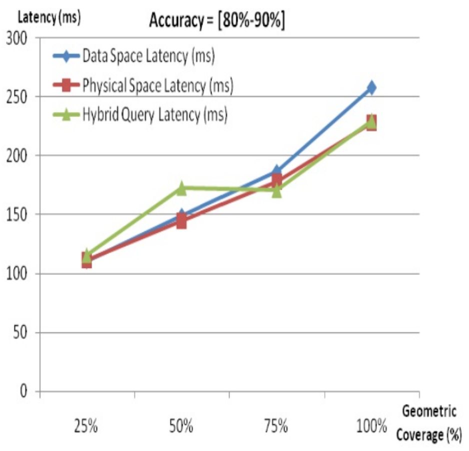


(b)

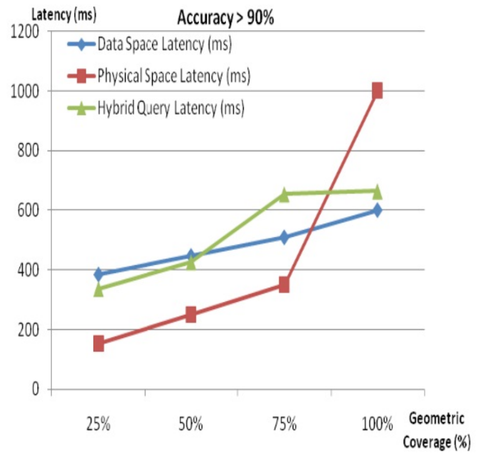


(c)

Figure. 16: Latency Vs. Accuracy corresponding to the geometric coverage of query as percentile of the area of the sensed field.

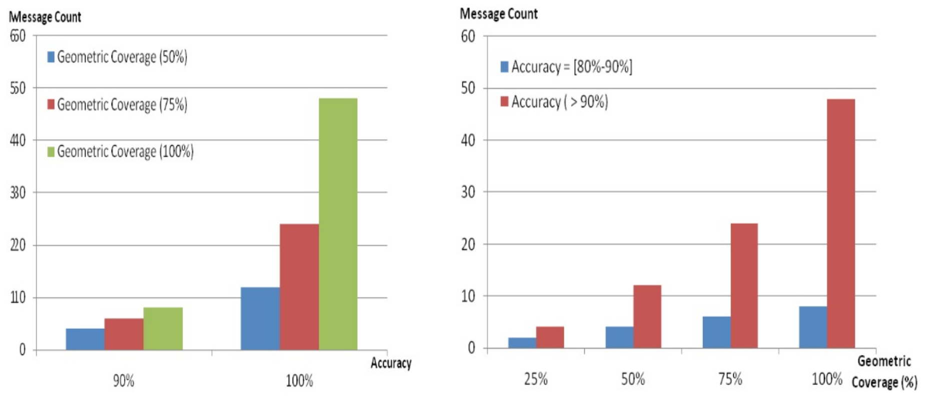


(a)



(b)

Figure. 17: Latency Vs. Query Coverage plot for accuracy = [80%-90%] & [90%-100%].



(a) (b)

Figure. 18: Number of messages communicated for different accuracies and geometric coverages.

well. 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 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 20 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 [Caicedo-Nuez et al. 2008]) 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.
The effect of mobility occurring simultaneously in multiple parts in the sensed field on the average mobility request latency is depicted in Figures 20 and 21. Figure 20 shows the result of mobility occurring in a set of non-neighboring, i.e, disconnected, clusters. In such scenario, only the nodes within these selected clusters are allowed to move according to the designated mobility model. Accordingly, some of these nodes will eventually exit their cluster, and join neighboring cluster(s). Once any of the neighboring clusters suffer from congestion it will request changing the border. The average mobility request latency intuitively increases with the increase in the number of clusters that have mobile sensor nodes. Fig. 20. Average Mobility Request Latency for Simultaneous Disconnected Clusters. In Figure 21, the mobility allowance is not given to specific clusters, but rather to a set of square size regions inside the field. A region can overlap with more than once cluster. Hence, congestion can occur in any cluster in the field. The average mobility request latency in the regions mobility setup is slightly lower than the cluster mobility setup, because nodes exiting a region do not necessarily exit a cluster, which does not make a difference from a nodes management perspective for the indexing structure. In both setups the

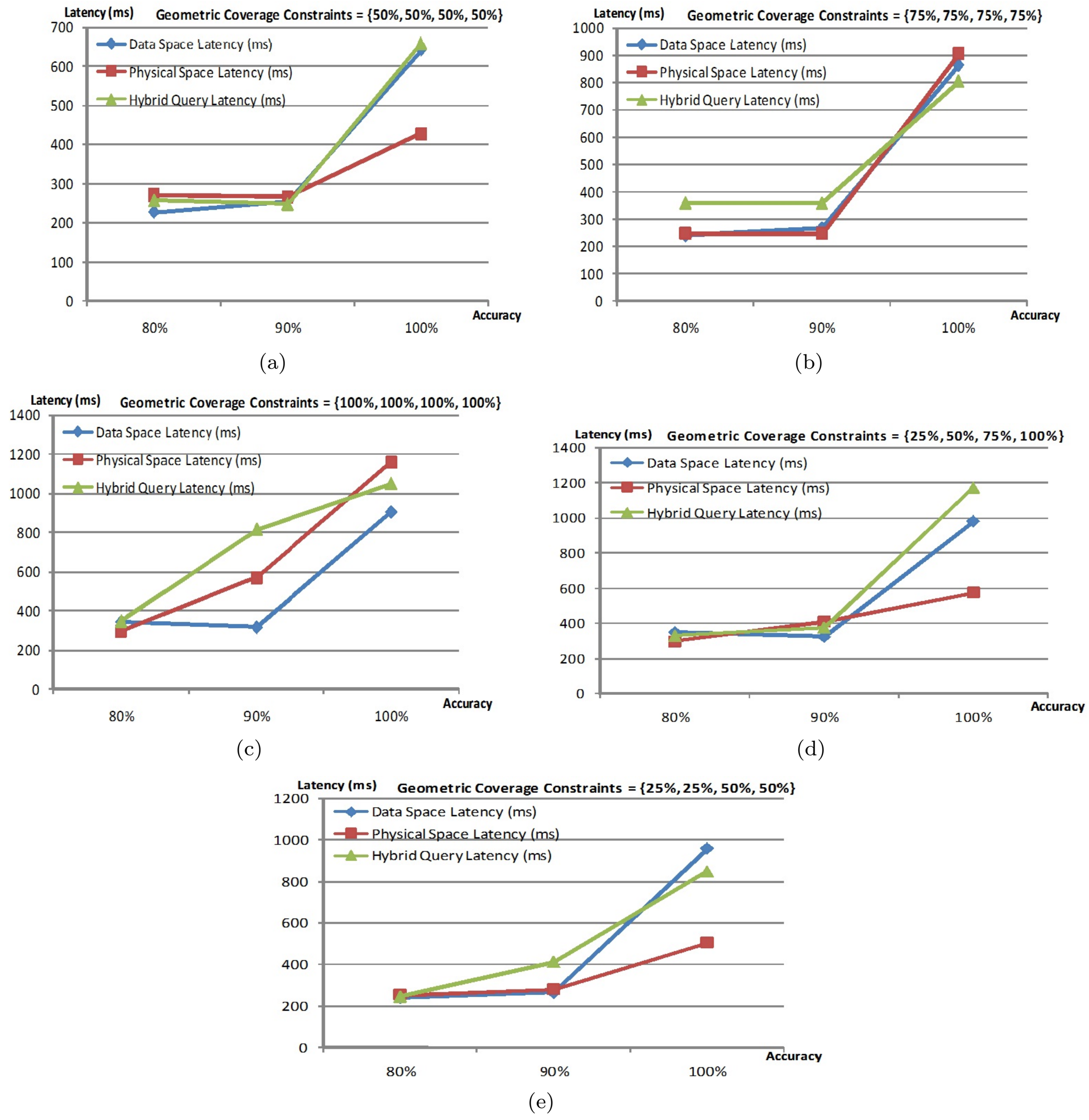


Figure. 19: Simulation Results for Queries Containing Different Query Coverage Constraints on Multiple Attributes.

random mobility model tends to have lower average mobility request latency. This is because of the random mobility nature which has less effect than the controlled motion, in the sense that a node can do at each step a motion decision that cancels the effect of previous steps. Therefore the overall mobility becomes less effective than the controlled mobility model, for which each motion step builds up towards a specified target. Fig. 21. Average Mobility Request Latency for Simultaneous Disconnected Regions.

In figure 21, 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%. 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 infor-

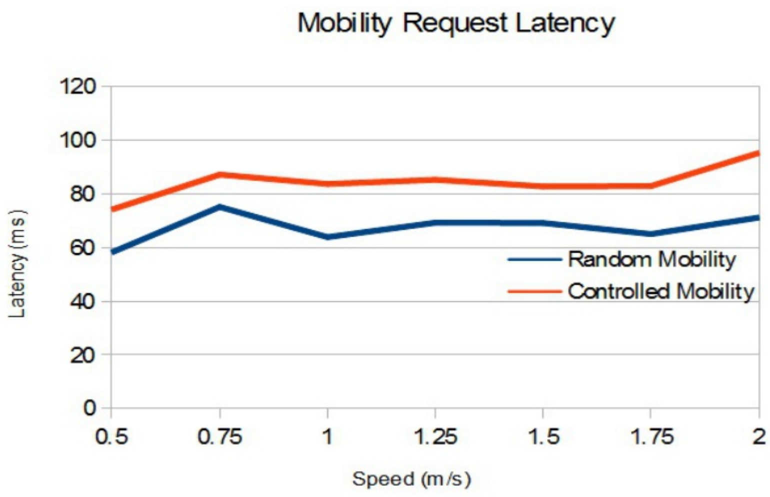


Figure. 20: Average latency of incorporating mobile node in the indexing structure Vs. sensor node speed.

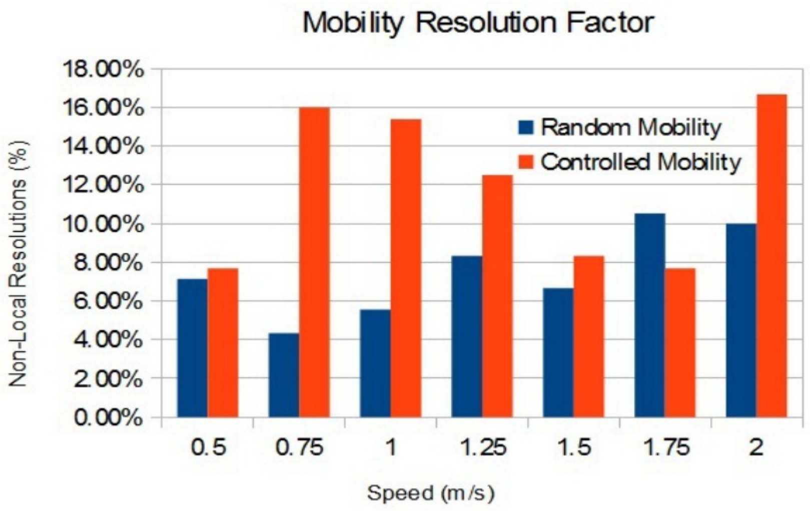


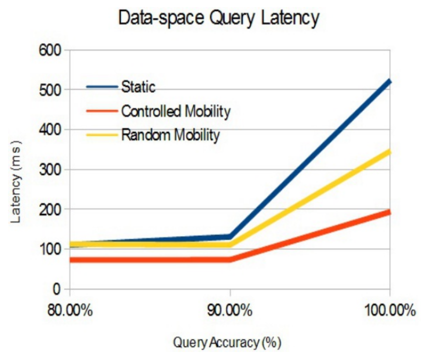
Figure. 21: Mobility Resolution Factor (MRF): The percentage of mobility requests that the mobility protocol is unable to resolve at the lowest level of the indexing structure.

mation traverse the whole indexing structure.

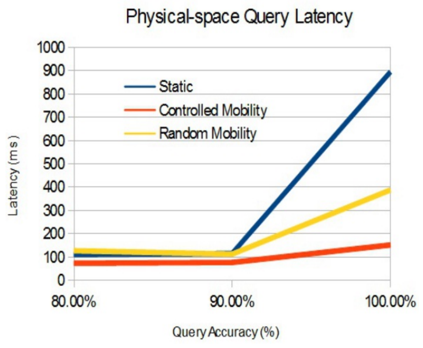
Figure 22 compares the latency of different data queries to the mobility managed structure (under random and controlled mobility) and the static structure where the indexing structure does not change itself to accommodate mobility and thus becomes relatively unbalanced. We present results for three different types of queries.

Figure 0?? 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 would have a higher memory footprint for the congested regions, which requires more processing and communication time. In figure 0??, 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.

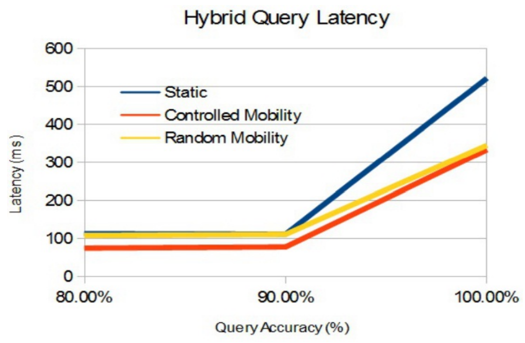
In figure 0??, the hybrid query latency can be viewed as a combination of latencies of both



(a)



(b)



(c)

Figure. 22: Query latency for (a) data-space, (b) physical-space and (c) hybrid queries Vs. required query response accuracy.

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

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 [Ciancio et al. 2006], 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 [Greenstein et al. 2003; Ganesan et al. 2005; Ouksel et al. 2007; Meliou et al. 2009; Zhang et al. 2003]. In [Greenstein et al. 2003], 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 [Greenstein et al. 2003; Ganesan et al. 2005], 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.

In [Meliou et al. 2009] an algorithm is proposed 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 [Gandhi et al. 2007]. 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 [Ouksel et al. 2007] 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 system doesn't support data-space indexing efficiently. Monitoring the WSN for events have been studied in [Liu et al. 2010], 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 [Zhang et al. 2003], a data dissemination scheme was proposed 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 [Su et al. 2011]. In [Liu et al. 2010], 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 [Liu et al. 2009], 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 [Gandham et al. 2003; Luo et al. 2005; Wang et al. 2005; Hanoun et al. 2008; Xing et al. 2012], in which the –one or multiple– sink node(s) moves in the field and gathers the sensed data. Non-hierarchical solutions, as [Gandham et al. 2003; Luo et al. 2005; Wang et al. 2005; Hanoun et al. 2008], study the optimal path to move across the field, in order to minimize latency. In [Xing et al. 2012], at two tier system of mobile sink node is proposed, 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.

8. CONCLUSION AND FUTURE WORK

In this paper we presented novel 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.
The proposed 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 proposed 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.
An efficient protocol is presented to manage and maintain in-network indexing structures in WSN under the constraint of mobile nodes. 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 proposed 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.
In our future work we plan to incorporate temporal dimension in the indexing framework to support spatio-temporal queries. We also plan to incorporate mobility models that involve higher mobility speed and uniform direction. In addition, we will study mobility management under higher dimensional indexing structures that do no involve orthogonal bisections. An extension of our work is to consider the mobility of the nodes participating in the indexing structure itself. Another extension is to incorporate the aspect of optimizing the coverage for multiple-events monitoring.

REFERENCES

Akkaya, K., and Younis, M. 2005. A survey on routing protocols for wireless sensor networks. In *Ad hoc networks, 2005. 3(3), 325-349*

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

Caicedo-Nuez, C.H., and Zefran, M. 2008. A coverage algorithm for a class of non-convex regions. In *Decision and Control, 2008. 4244-4249*

Chakrabarti, A., Sabharwal, A., and Aazhang, B. 2003. Using predictable observer mobility for power efficient design of sensor networks. In *Information Processing in Sensor Networks. Springer Berlin Heidelberg 129-145.*

Chen, J., Johansson, K. H., Olariu, S., Paschalidis, I. C. and Stojmenovic, I. 2011. Guest editorial special issue on wireless sensor and actuator networks. In *IEEE Transactions 2011. 56(10), 2244-2246.*

Chui, C. K. 1992. An introduction to wavelets. In *Academic press.*

Ciancio, A., Pattem, S., Ortega, A., and Krishnamachari, B. 2006. 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. 309-316.*

Dantu, K., Rahimi, M., Shah, H., Babel, S., Dhariwal, A., and Sukhatme, G. S. 2005. Robomote: enabling mobility in sensor networks. In *Proceedings of the 4th international symposium on Information processing in sensor networks. IEEE Press 55.*

Dietrich, I., and Dressler, F. 2009. On the lifetime of wireless sensor networks. In *ACM Transactions on Sensor Networks (TOSN), 5(1) 5.*

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

Gandham, S. R., Dawande, M., Prakash, R. and Venkatesan, S 2003. Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In *Global telecommunications conference, 2003. GLOBECOM'03. IEEE (Vol. 1, pp. 377-381).*

Gandhi, S., Hershberger, J., and Suri, S. 2007. Approximate isocontours and spatial summaries for sensor networks. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on (pp. 400-409).*

Ganesan, D., Greenstein, B., Estrin, D., Heidemann, J., and Govindan, R. 2005. Multiresolution storage and search in sensor networks. In *ACM Transactions on Storage (TOS), 1(3), 277-315.*

Ghica, O. C., Trajcevski, G., Scheuermann, P., Bischof, Z., and Valtchanov, N. 2008. Sidnet-swans: A simulator and integrated development platform for sensor networks applications. In *Proceedings of the 6th ACM conference on Embedded network sensor systems (pp. 385-386).*

Greenstein, B., Ratnasamy, S., Shenker, S., Govindan, R., and Estrin, D. 2003. DIFS: A distributed index for features in sensor networks. In *Ad Hoc Networks, 1(2), 333-349.*

Hanoun, S., Creighton, D., and Nahavandi, S. 2008. Decentralized mobility models for data collection in wireless sensor networks. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on (pp. 1030-1035).*

Hosking, J. R. 1984. Modeling persistence in hydrological time series using fractional differencing. In *Water resources research, 20(12), 1898-1908.*

Keshner, M. S. 1982. 1/f noise. In *Proceedings of the IEEE, 70(3), 212-218.*

Liu, C and Cao, G. 2010. Distributed monitoring and aggregation in wireless sensor networks. In *INFOCOM, 2010 Proceedings IEEE (pp. 1-9).*

Lou, J., and Hubaux, J. P. 2005. 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 (Vol. 3, pp. 1735-1746).*

Meliou, A., Guestrin, C., and Hellerstein, J. M. 2009. Approximating sensor network queries using in-network summaries. In *Information Processing in Sensor Networks, 2009. IPSN 2009. International Conference on (pp. 229-240).*

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

Mohamed, M.M.A, Khokhar, A.A, and Trajcevski, G. 2013. Energy efficient in-network data indexing for mobile wireless sensor networks. In *Advances in Spatial and Temporal Databases (pp. 165-182). Springer Berlin Heidelberg.*

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

Ouksel, A., Xiao, L., and Hauswirth, M. 2007. Dynamically Self-Organizing Sensors as Virtual In-Network Aggregators and Query Processors in Mobile Ad-Hoc Sensor Databases. In *2007 IEEE 23rd International Conference on Data Engineering (ICDE 2007).*

Pileggi, S. F., Fernandez-Llatas, C., and Meneu, T. 2011. Evaluating mobility impact on wireless sensor network. In *Computer Modelling and Simulation (UKSim), 2011 UkSim 13th International Conference on (pp. 461-466).*

PON, R., BATALIN, M. A., GORDON, J., KANSAL, A., LIU, D., RAHIMI, M., AND ESTRIN, D. 2005. Networked infomechanical systems: a mobile embedded networked sensor platform. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on (pp. 376-381).*

SAMET, H. 1990. The design and analysis of spatial data structures. In *(Vol. 85, p. 87). Reading, MA: Addison-Wesley.*

SHI, H. AND SCHAEFFER, J. 1992. Parallel sorting by regular sampling. In *Journal of Parallel and Distributed Computing, 14(4), 361-372.*

SOMASUNDARA, A. A., RAMAMOORTHY, A., AND SRIVASTAVA, M. B. 2004. Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In *Real-Time Systems Symposium, 2004. Proceedings. 25th IEEE International (pp. 296-305).*

SUSHANT, R. C. S. S. R.. Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks.

WANG, Z. M., BASAGNI, S., MELACHRINOUDIS, E., AND PETRIOLI, C. 2005. Exploiting sink mobility for maximizing sensor networks lifetime. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on (pp. 287a-287a).*

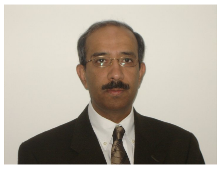
WILLINGER, W., TAQQU, M. S., SHERMAN, R., AND WILSON, D. V 1997. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level. In *Networking, IEEE/ACM Transactions on, 5(1), 71-86.*

XING, G., LI, M., WANG, T., JIA, W., AND HUANG, J. 2012. Efficient rendezvous algorithms for mobility-enabled wireless sensor networks. In *Mobile Computing, IEEE Transactions on, 11(1), 47-60.*

ZHAO, F., AND GUIBAS, L. J. 2004. Wireless sensor networks: an information processing approach. In *Morgan Kaufmann.*

http://jist.ece.cornell.edu/index.html. *[Online]*

Mohamed M.Ali Mohamed received his B.S. in Computer Engineering from Cairo University, Egypt, in 2007. In 2008 he joined the University of Illinois at Chicago, where he received his M.S. in Mechanical Engineering in 2009 and, recently, his Ph.D. in Electrical and Computer Engineering in 2015. His main research interests are in the areas of distributed algorithms, spatio-temporal data management in wireless sensor networks,and high performance computing. Over the course of his Ph.D. work, Dr. Mohamed published several papers in refereed conferences, and received a Best Short Paper Award at the ACM MSWiM conference (2013). His research work was funded by National Science Foundation grants. His industry experience is in the photo-lithography and electronic design automation industries, where he worked at the world leading corporations and has been contributing to their cutting-edge products.



Ashfaq Khokhar is serving as a Professor and Chair of the Department of Electrical and Computer Engineering since Fall 2013. Dr. Khokhar joined Illinois Institute of Technology from the University of Illinois at Chicago (UIC), where he was a Professor and Director of Graduate Studies in the Department of Electrical and Computer Engineering. He also held adjunct appointments in the Department of Computer Science and the Department of Biomedical and Health Information Sciences at UIC.
Dr. Khokhar earned a B.S. in Electrical Engineering from the University of Engineering and Technology in Lahore, Pakistan, in 1985; an M.S. in Computer Engineering from Syracuse University in 1989; and a Ph.D. in Computer Engineering from the University of Southern California in 1993. Dr. Khokhar is also the chief technology officer of Video Analytica, Inc., a Woodridge, Ill.-based startup company he founded that develops scalable and computationally efficient solutions for multimedia applications, including smart and intelligent video surveillance technologies.
Dr. Khokhars research centers on context-aware wireless networks, computational biology, health care data mining, content-based multimedia modeling, retrieval and multimedia communication, and high-performance algorithms. He is considered a leading expert in the area of high-performance solutions for multimedia applications, especially those that are data or communication intensive. Under his directorship, the research group at UICs Multimedia Systems Lab has grown substantially. His lab has graduated more than 20 Ph.D. students in the last 10 years.
Dr. Khokhar has contributed to five edited volumes and co-authored nine book chapters, 55 publications in archival journals, and 158 refereed conference papers. Additionally, he has presented his work and has served as program chair at several prominent conferences. His research is currently supported by multiple awards funded by the National Science Foundation (NSF) and National Institutes of Health. In the recent past, Khokhars research has also been supported by the United States Army, the Department of Homeland Security, and the Air Force Office of Scientific Research. The funding level associated with these activities has been well over $10 million to date.$
Dr. Khokhar has extensive teaching experience and is highly committed to both graduate and undergraduate education. Over the years, he has developed new graduate courses in the areas of parallel computing and multimedia systems. At UIC, he revised the contents of several existing graduate and undergraduate courses to adapt to the rapid changes in the field and to reflect the latest research trends. Dr. Khokhar has actively pursued reorganization of the undergraduate curriculum in the computer engineering and computer science areas and has collaborated with his colleagues to streamline the contents of different courses and link them to state-of-the-art trends in the computer industry.
Dr. Khokhar has received numerous scholarly recognitions and awards, including the NSF CAREER Award in 1998 for his work on multi-threaded algorithms for multimedia applications. He has served as associate editor on the editorial board of numerous journals. Khokhar was elected an Institute of Electrical and Electronics Engineers (IEEE) Fellow in 2009 for his work on multimedia computing and databases.

Goce Trajcevski received his B.Sc. degree from the University of Sts. Kiril i Metodij, and his MS and PhD degrees from the Dept. of Computer Science at the University of Illinois at Chicago. His main research interests are in the areas of spatio-temporal data management, routing and data management in wireless sensor networks, and reactive behavior in dynamic systems. He has published over 90 papers in refereed conferences and journals and received a Best Paper Award at the CoopIS conference (2000), Best Paper Award at the IEEE MDM conference (2010) and Best Short Paper Award at ACM MSWiM conference (2013). His research has been funded by BEA, Northrop Grumman Corp., NSF and ONR. He has served as an associate editor at ACM DiSC, and is presently an associate editor of GeoInformatica and ACM Transactions on Spatial Algorithms and Systems (TSAS). He has been part of organizing and program committees in numerous conferences and workshops, and serves as an Associate Editor for the GeoInformatica journal and the ACM Transactions on Spatial Algorithms and Systems. Presently, he is an Assistant Chairman with the Department of Electrical Engineering and Computer Science at the Northwestern University.