EVALUATING INDEXING AND ROUTING SCHEMES FOR SPATIAL QUERIES IN SENSOR NETWORKS

by

RAJA RAJESHWARI ANUGULA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2006

ACKNOWLEDGEMENTS

I would like to thank my supervising professor, Dr. Ramez Elmasri, for his guidance and support, and for giving me an opportunity to work on this thesis. I appreciate his insights and suggestions, which led to the successful completion of this work. I would also like to thank Dr. Yonghe Liu and Dr. Hao Che for their time and valuable suggestions.

I owe my sincere thanks to my Korean friends for their constant support and encouragement throughout this thesis. I would like to thank my friends for their support.

I am thankful to my parents and my love for their support and encouragement throughout my academic career.

December 21, 2006

ABSTRACT

EVALUATING INDEXING AND ROUTING SCHEMES FOR

SPATIAL QUERIES IN SENSOR NETWORKS

Publication No. _____

Raja Rajeshwari Anugula, MS CSE

The University of Texas at Arlington, 2006

Supervising Professor: Dr.Ramez Elmasri

Recent advances in low-power sensing devices coupled with widespread availability of wireless ad-hoc networks, has fueled the development of sensor networks. These senor networks have various applications such as to monitor conditions at different locations (temperature, pressure, rainfall, vibrations etc.,), tracking of objects and so on. Each device is equipped with an energy source (usually with a battery), memory, CPU and communication bandwidth, which is severely constrained. Hence each sensor network is comprised of hardware for sensing, software for communication and computational algorithms. Spatial queries are commonly applied to sensor network, for example: “Find the highest temperature sensed in a particular

region". Spatial query processing is considered extensively in the context of centralized databases. In this thesis, we examine spatial queries in distributed sensor networks. We use a binary tree, which is constructed by a MULT routing protocol and the meta-data that has been collected in the base station to construct an index structure, thus resolving the spatial queries, we evaluate and compare the behavior of our approach with other kind of indexing structure and show that our mechanism is efficient in terms of increased in-network computation and lowered communication.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Table Page

CHAPTER 1

INTRODUCTION

Recent advances in wireless communications coupled with efficient query processing have increased the ubiquitous applications of Wireless Sensor Networks (WSN). Sensor Network technology is increasingly applied to the research applications in the world of Ad-hoc networks Sensor nodes are small, interconnected, battery operated, units, which are usually less mobile. They are more densely deployed than the ad-hoc networks [1]. In General, these sensors have finite radio range, limited computational power and less communication bandwidth. The major activity of these sensors in the sensor field is to collect data and transmit them, possibly compressed and/ or aggregated with those of the neighboring nodes to the other nodes or to the sink. These are randomly deployed in tens and thousands to gather physical data for the variety of purposes such as environmental monitoring, surveillance of the military area, etc., The typical application is to monitor a particular geographical area over a time period or to gather data such as temperature values, movements of the enemy. The sensor data is obtained by supplying desired query to the network topology. Among those are the spatial queries which are a subset of queries in which the database or the sensor

network is queried by location rather than an attribute. Spatial queries are used to answer the queries such as “Find the Maximum temperature in Arlington area”. In traditional Spatial database we have many spatial indexing techniques such as R tree, R+ tree and R * tree [2, 3, 4]. But the data in these techniques are centralized and the indexing is performed on them to execute the spatial query. Where in Sensor network the data is distributed and the queries also have to be processed in distributed manner. Because of the resource limitation of the sensor nodes, it is desirable to process the query on those sensors, which has the relevant data.

1.1 Overview

Centralized indexing techniques like R-tree and Quad tree provide a fast mechanism in traditional database. These techniques create the spatial index on the objects or the group of objects, which are co-related geographically and implement it to process the spatial query. But in the case of sensor network, it differs because of the characteristic of the sensed data, i.e. what kind of measurement it majors in, as well depends on the durability of the sensor i.e., a sensor may at times fail or join the network. The index should keep track of all the networks, which participate in answering the query. Such kind of unique characteristics of the sensor networks generate new challenges for processing spatial queries in a typical settings of WSN.

a) Execute the query in distributed fashion: Because the sensor data is distributed, the queries must be executed in the distributed manner.

b) Concept of Distribution: The energy cost of communication is far more than the computation. This is optimized, by the decision of where to run the query in the network

c) Index Structure: The sensors may fail or join the network, thus the network topology and the supported index has to re-organize it self.

Although collecting spatial query results in a spatially enabled sensor network is same as collecting attribute results. With the fact that, the sensors that are located in the same geographical area yields significant reduction of energy consumption in processing the spatial query.

1.2 Motivation

Our work is motivated by the above facts that and on the decision of which few subset of nodes could answer that are related to the query, instead of submitting to the whole area. Employing the previous work MULT, we tried to develop the indexing structure using the meta-data designed in the base station and compare it with the other historically used spatial indexing structure such as R-tree and Quad tree. The Base station which is generally considered as the unlimited power has the ability to store and able to manage many tedious calculations. In this work we considered the base station to be equipped with GPS to know the spatial positions of each Sensor nodes. Thus, maintaining a tabular data for the lookup of node information. Hence, facilitating the decision of which subset of nodes is required to answer the query preventing the remainder nodes from redundant expenditure of the energy.

In this thesis we evaluate indexing and routing schemes by processing the spatial queries in distributed fashion. We exercise MULT protocol to maintain the decentralized spatial index when sensors fail or join the network. We study the effect of using this mechanism with traditional indexing methods and perform an extensive set of experiments to illustrate the efficiency, scalability and performance of our approach.

Our contribution towards this thesis:

- We build a distributed indexing structure over the sensor network which is maintained by the base station to efficiently evaluate the spatial queries
- We present a decentralized way of constructing supportive indexing tree aided with a routing protocol in the sensor network.
- We present a distributed way of optimizing the spatial index to reduce energy consumption of the system for spatial queries
- We perform extensive experiments on this spatial index structure in Cartesian co-ordinates, in sparse and dense networks. And evaluate, on our mechanism and on the traditional methods.

The rest of the paper is organized as follows. Section 2 reviews related existing work. Section 3 gives the scenario of our architecture for sensor networks. Section 4 proposes the routing model and the extended indexing structure. Section 5 discusses the implementation and evaluation of ABT. Section 6 gives the summary of experimental results. Finally, we conclude and discuss future direction of this work.

CHAPTER 2

RELATED WORK

2.1 Spatial Indexing Techniques

2.1.1 R-Tree

Spatial Query processing has been widely analyzed in the quarters of centralized data base systems. R-tree [2] is one of the most preferred methods for indexing spatial data. In R-tree, a spatial database consists of collection of tuple that has a unique identifier which can be used to retrieve it. The Spatial data objects are grouped using minimum bounding rectangle (MBR). Objects are added to an MBR within the index that will lead to smallest increase in its size. Each entry within a non-leaf node stores an n MBR that covers all the MBR's in the children nodes. Leaf nodes in an R-tree contain index record entries of the form two pieces of data, (ptr, rect) where ptr is pointer to the object in the database and rect is the MBR of the object. The searching algorithms range from simple such as exhaustive, Linear, Quadratic to the complex, R*, Hilbert packed/non-packed, Morton Packed/Non-Packed. These help in the decision of searching down the sub-tree of the child node. Variants of the R-tree structure such as R+ tree and R * tree [2, 3] have also been proposed.

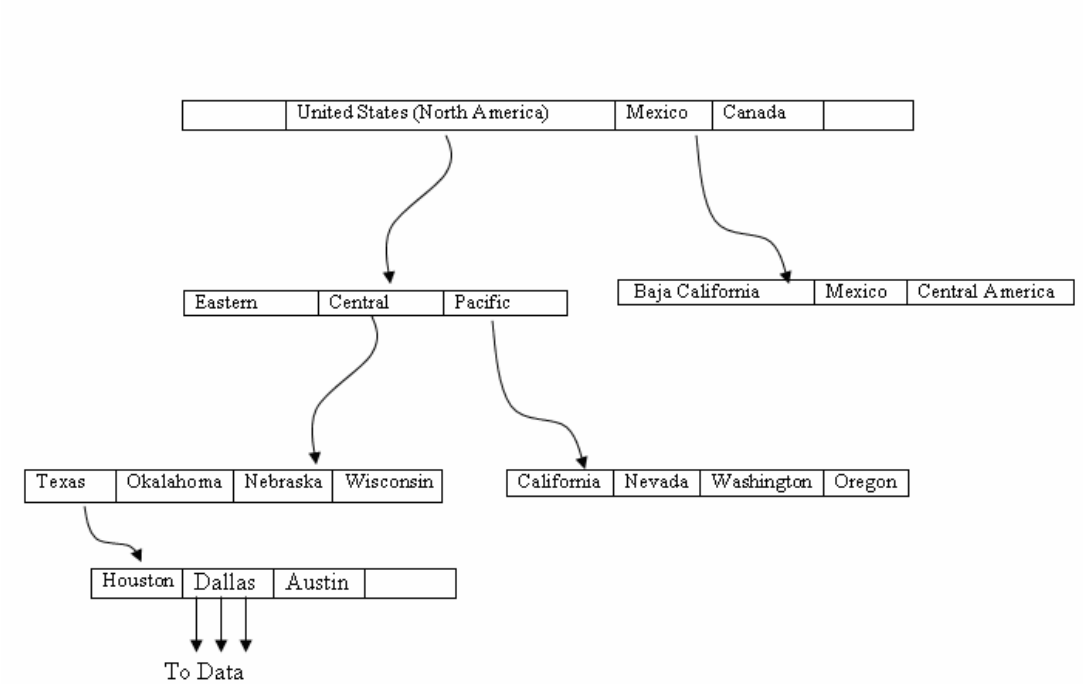

Figure 2.1 Structure of the R-tree

2.1.1.1 Description of the figure 2.1

The figure 2.1 above shows the structure of the R- tree, where the regions (rectangles) are stored as index, and pointers to these are used to access or traverse the internal nodes. The data structure splits the space with hierarchically nested and overlapping boxes. Each region is comprised of X, Y co-ordinates of geographical data .The search on this indexing structure starts from the root node, considering each rectangle as an input. The internal nodes contain set of rectangles and pointers to corresponding child nodes. Every leaf node contains the rectangles of spatial objects. This kind of searching is similar to B-tree.

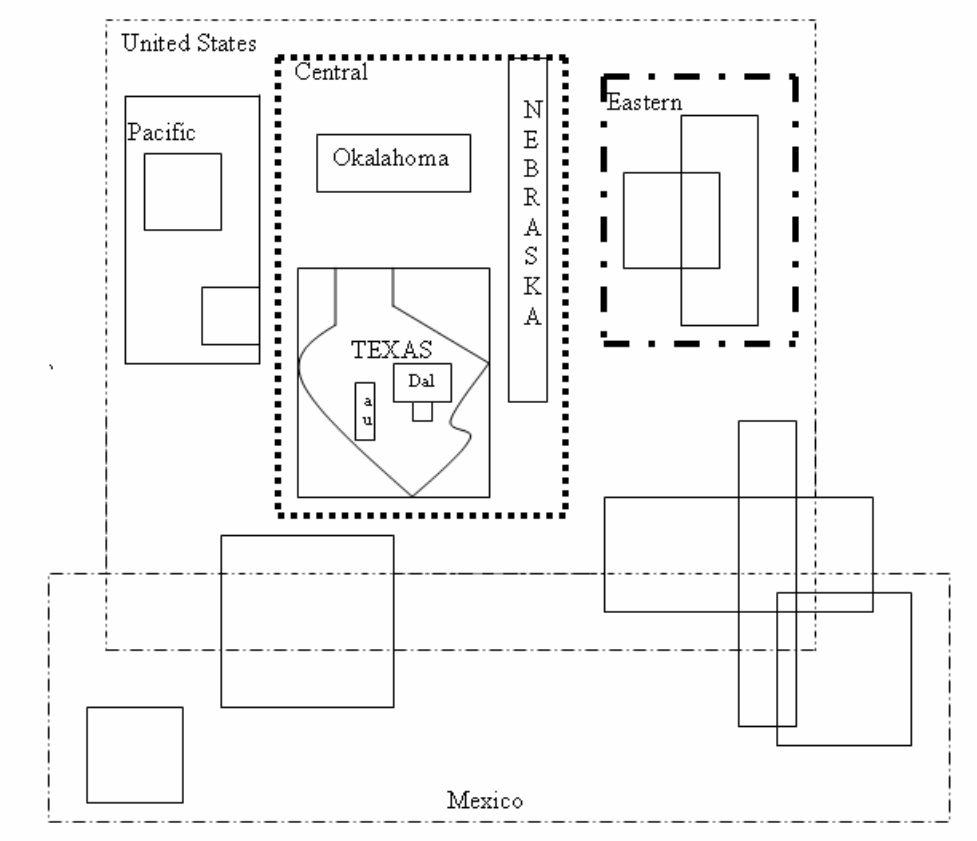

Figure 2.2 Intersecting and overlapping relationship between the rectangles

2.1.1.2 Description of figure 2.2

In the above figure 2.2 each rectangle is an input. For every rectangle in the node, it has to be decided if it overlaps the search rectangle or not. If yes, the corresponding child node also has to be searched. For example, in Fig 2 to search "Dallas" rectangle intersecting with the query window, we need to process the "Central" rectangle among all the rectangles that are formed under United States. Then the rectangle "Texas" needs to be traversed to reach all the way to "Dallas" node. Thus, in this way all the overlapping nodes have to be traversed recursively.

2.1.2 Quad Tree

One another such kind of spatial tree is Quad tree which describes a Hierarchical data structures whose common property is that they are based on the principle of recursive decomposition of space. They are actually differing in two ways -1) the type of data that are used to represent, 2) the principle guiding the decomposition process. Currently we consider the point data (i.e. the sensor data) as well as the region data (aggregated).And the decomposition is termed as regular square tiling, generally, it is dynamically governed by properties of the input i.e. placement of the each sensor node

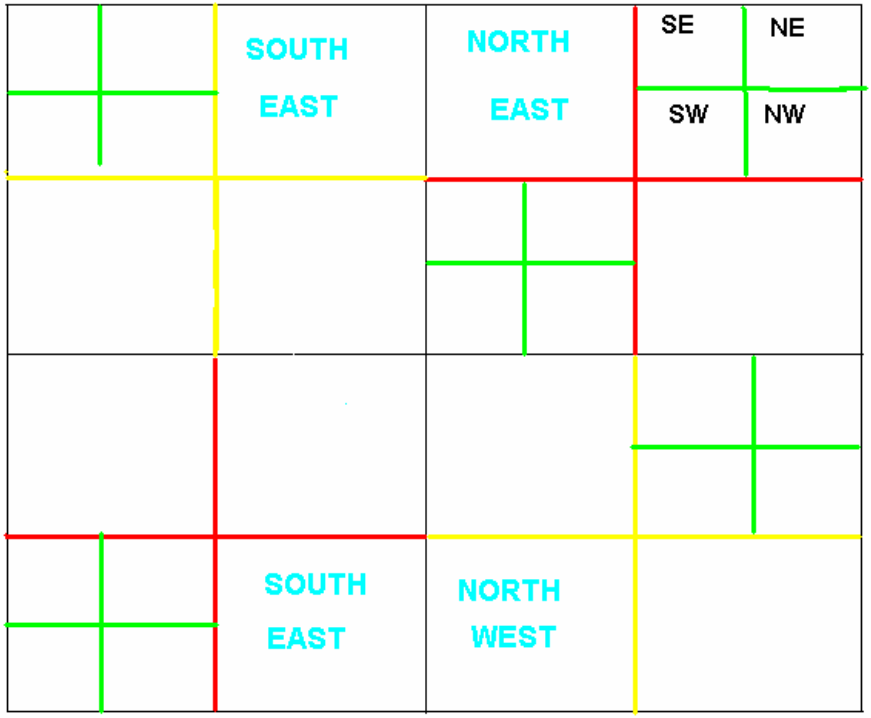

Figure 2.3 Successive divisions into sub quadrants (NE, NW, SE and SW)

2.1.2.1 Description of the Region Quad tree

The Quad tree in this section we refer will be the most studied Quad tree which is concerned with the representation of Region data. This is based on the successive subdivision of the space portioning each time into four equal size quadrants. It is then subdivided into quadrants, sub quadrants etc. Each node is represented by the node-id (derived from its parents) and an offset pointing to another node id, if it is a non-leaf node or to the data item, if it is a leaf. Morton [1966] used it as a means of indexing into a geographic database

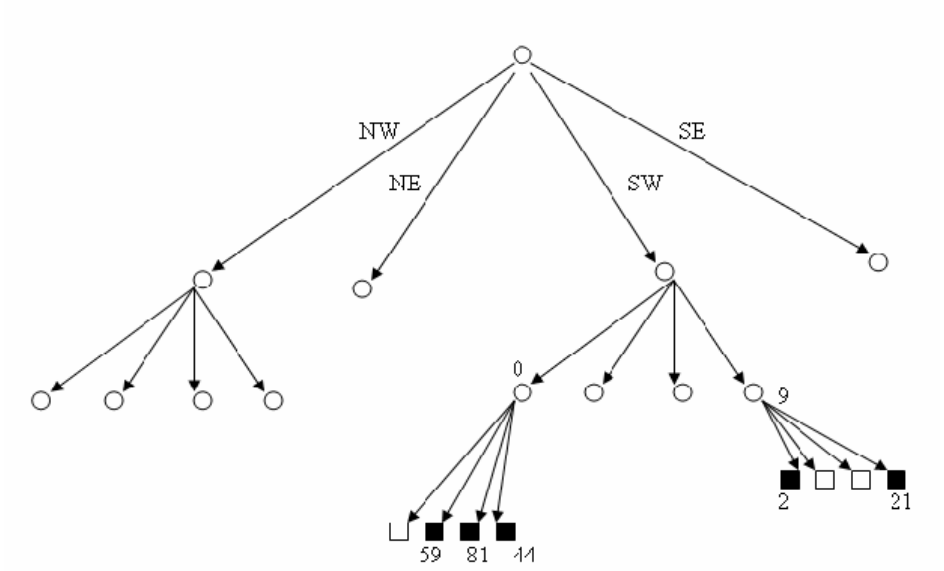

Figure 2.4 Construction of Quad tree

2.1.2.2 Description of quad tree

The fig 3, shows the rectangular grid. At the first level is evenly divided into four quadrants North East, North West, South East, and South West. Each sub-quadrant is treated as child node and thus we have four children. In the second level, these are again successively divided into 4 equal quadrants NE, NW, SE, and SW. So,

16 sub-quadrants decomposed four for each in the previous level as shown in fig 4. Finally the leaf nodes are the blocks where the data is stored. In order to index, we need to traverse the index tree as per the four directions. Thus the search is made easier with each successive subdivision of the space into 4 quadrants.

2.1.3 Some other novel tree structures

Range queries have been studied in dynamic and large-scale sensor environments [10, 17, and 12]. Building a centralized index, a distributed index or a super peer-network is not practical in sensor networks due to high resource limitation on the sensor nodes and the sensor field. Ferhastosmanogulu et al [16] has proposed a tree, in which they partition the sensor network into hierarchical shaped clusters. Their techniques implement join/splits, whenever the nodes join or die out, the cluster reorganize it self and constraints itself to a certain criteria satisfying each cluster. They came up with this novel structure called peer-tree, which is used to answer Nearest Neighbor queries. In [3] the peer tree is created bottom up fashion by joining together the group of objects that are close to each other. Each group selects a representative, which acts as parent. As a result, the connections between parents and children becomes progressively longer, assuming the nodes transition range is in the order of the dimensions of the sensor networks. Our technique on the other hand operates in a top down fashion while constructing the hierarchy and guarantees one hop from any parent to its child.

2.2 Routing Techniques

2.2.1 Discussion of Different Types of Routing Scheme

In this thesis, we advocate a database approach reinforced by a routing protocol, in a sensor network. Since nodes are usually powered by batteries, increasing network design is the major goal of any sensor network system. Data transmission back to the central storage for offline storage, querying and data analysis is very expensive for large size sensor networks, because it consumes lots of energy. Since sensor node have local computation abilities, the In-Network aggregation methodology can significantly improve the network lifetime. Directed communication [15] forwards the query to the whole network and in return each sensor sends its data directly to the base station. If the base station is far away from the nodes, direct communication will require a large amount of transmit power from each nodes. This will quickly drain the battery of nodes and reduce the system lifetime. There are many power aware techniques that have been proposed to reduce energy usage in sensor networks LEACH [13] proposes an energy adaptive efficient clustering protocol that uses randomization to distribute energy load evenly among the sensors in the network. Hu et al [5] present a proactive caching scheme for mobile environment. Their idea is to create an index (R-tree and a cache for the spatial query results and use the cached scheme for mobile base station). The number of requests that need to be send to the base station. Our technique on the other hand not only insight the energy and load balance, but also reduce the size of the queried area. Unlike the above-mentioned techniques we design our spatial index in a way that it can be applied to sensors with limited resource usage for processing the spatial queries.

Many routing protocol have been proposed to route a packet to a specific location in the sensor network. Directed Diffusion [20] forwards the request based on the user interest such as location. Geographic based routing [6] [7] use geographic co-ordinates to route queries to the specific node. Unlike the general routing protocols we focus on running spatial queries that query the sensor network for a specific area. In our approach we built a distributed spatial index over the sensor network and at the same time reducing the energy consumption disseminating and processing the spatial queries.

Several attribute based query processing have been developed for sensor networks. Madden et al [5] [6] have proposed Acquisitional Query processing (ACQP) executes attribute queries over sensor network. TAG [6] exploits the declarative queries over sensor networks. ACQP builds a semantic routing tree (SRT) that is conceptually an attribute index on the network. It stores a single one-dimensional interval representing the range values beneath each of its children in each node. Every time a query arrives at a node n, n checks to see if its child's value overlaps the query range. In our work for next section 4 we introduce the MULT - adaptive Binary tree which helps storing all the MBR's area formed in the Sensor network, the parent and child entries, node position and cluster head or a not. Our approach manages the minimum bounding rectangle area to obtain better performance in spatial query execution while saving sensor resources.

The proposed work is a decentralized approach of executing spatial queries in sensor network, without any assumption of the capabilities of the sensor nodes. We focus on

processing spatial queries in sensor network while exercising a protocol called MULT [18] that reduces the network energy consumption processing the queries.

2.3 Spatial Queries over a sensor network

Spatial queries are used to answer questions such as "What is maximum temperature in the Area X?" Basically, spatial query processors execute spatial queries in two ways: a) Finding the number of sensors in the rectangle b) Filtering out the sensors that do not satisfy the spatial constraint.

A spatial query has one or more spatial constraint, which represents the area of interest such as calculating all the si which satisfies the spatial query constraint -belongs to query area q. In this thesis we consider, Aggregation Spatial queries such as SUM, MIN, MAX, and AVG, applied to a set of values (sensor data values).A spatial query in a sensor network is a function F $\{d_i \mid L_i \in R\}$, where $d_i \in R$ is a value and a location si $\in R^2$ (the values are real and the locations are x,y coordinates) .R is a range of the form [a, b] X [c, d], (a, b, c, d $\in$ R i.e they are Real numbers) and the sensors in the area when it's x-coordinate is between a and c and the y coordinates are between b and d. This allows finding and/or aggregating sensor data located in the defined area of interest such as window, circle, polygon or trace. And the nodes are filtered out if they do not satisfy the spatial constraint.

CHAPTER 3

ARCHITECTURE OF SENSOR NETORKS

3.1 Network Model

A sensor network consists of a large number tens and thousands of sensor nodes. Individual sensor nodes are connected to other nodes in their radio range through wireless communication interface, and they use a multi-hop routing protocol to communicate with other sensor nodes, which are spatially distant. Nodes also have limited computation and storage capabilities. A node has a general purpose CPU to perform computation and a small amount of storage space to save the results ad data. Since nodes are usually not connected to fixed infrastructure they use batteries as their main power supply and preservation of power is one of the main goals of the sensor networks. Since communication consumes more energy than computation.

In our thesis we consider a situation where sensor nodes might reside in a hostile environment, such as battlefield or in a region of recent disasters to support rescue missions. Thus, posing us the situation of join/failure of the nodes in the sensor field. Eventually the data being sensed by the nodes in the network must be transmitted to a base station where the end user can access the data. There are many possible models for these micro sensor networks. In this work we consider we assume the following situation:

1. The base station is fixed and located far from the sensors.
2. All nodes in the network are homogeneous and energy constrained.
3. Base station is not energy constrained.
4. Sink node is one among the sensor nodes and is one hop from the base station.
5. Each sensor can perform multi-measurement such as temperature, humidity, rainfall etc.

3.2 Architecture for Sensor database system

Previously, sensor networks work mainly as data collectors and transfer data from sensor nodes to the base station, where the data is aggregated and stored for offline querying and analysis. Fjords improve the centralized architecture by sharing scan operators at the sensor nodes and switching sensors on and off according to query specifications [23] they help to reduce energy consumption, as nodes are aware of the queries , but they lack support, for more advanced query processing techniques[23]. Since local computation is much cheaper than computation, pushing partial computation out onto the network could significantly improve energy efficiency. We propose loosely coupled distributed architecture to support both computation and a generic in-network aggregation.

In this thesis, we built spatial index to maintain a spatial database in the base station. We assume the base station is aware of the location of all sensors which even join/fail in the network through MULT routing protocol, which we introduce in the section 5. Hence these spatial index process spatial queries in distributed fashion.

Thus, reducing the complexity of processing the spatial query processor running on each sensor uses this index structure to

1. Bound the MBR's, which do not lead to any result.
2. Find a path to the sensors that might have a result
3. Aggregate the data in the network to reduce the number of packets transferred and save energy.

We describe the construction of this index and routing structure in the next section

3.3 In – Network aggregation

In this thesis, we setup the WSN with hundreds and thousands of sensor nodes that are deployed in randomized or arbitrary manner. The component of power consumption is in sensing, communication, and data processing. Among these, the communication through out the sensor networks and in reaching the base station consumes the principal amount of energy. For this reason, in our application scenario, we try to obtain an aggregated multi-hop data transmission instead of direct communication of sensed raw data to the base station. This poses us the challenge of processing and combining data near the node where it is generated. So we implement processing and combining data in the network. This is possible with the idea of internal aggregation [20]. The Generic view of wireless sensor network in two dimensional Cartesian co-ordinate spaces is reflected as a base station situated outside of the network and large number of sensors is deployed in random manner. One of the nodes, which are at the border of the sensor grid, will be nominated as the sink nodes. All the sensors know location of each other with the aid of co-ordinate

system and the built in radio trans-collector. Fig 5 shows the data aggregation in the network. In this thesis, we consider Query based approach. For example, if the user sends the query through the base station and seeks the summarized view of the area of interest.

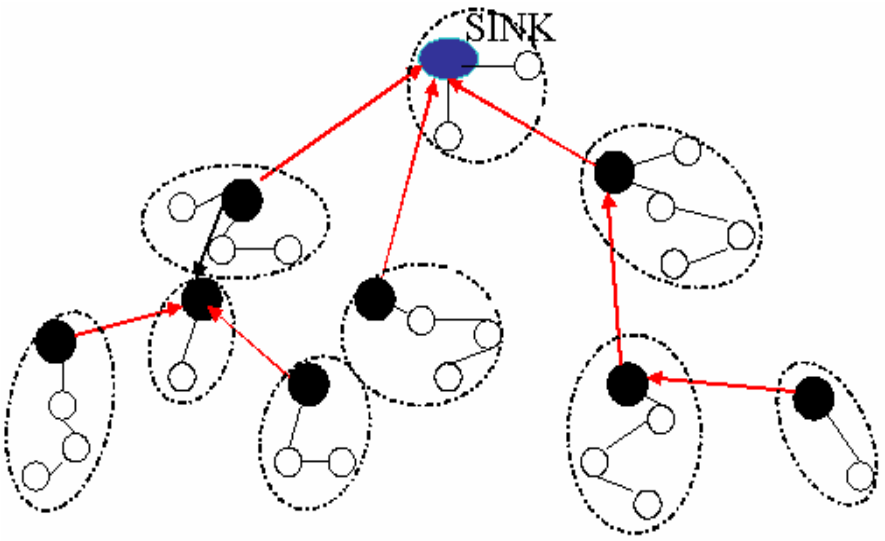

Figure 3.1 Concept of In-Network Aggregation

The different properties of aggregation make the computation more effective for processing raw data into meaningful information for the provided query. Hence with least amount of work, the simple computation makes aggregation an important concern for the sensor nodes. In our approach wee study to compress aggregated data in the following steps:

- The sensor data is the raw data stocked by each individual sensor nodes, which is communicated through multiple hops for collective processing at each parent node.

- The data from the child nodes in addition with data sensed at the parent node is called the aggregated data .This whole data is send to the head node of its respective cluster (MBR).

Each sensor carries out the task of sensing, for the requested query and collectively performs the aggregation function, then communicating back to the user. The commonly used aggregated functions are MAX, MIN, AVG, SUM and COUNT. The aggregation is performed only at few intermediate nodes. The aggregated result of each cluster is held by the cluster head. The strategy of nomination of each cluster head will be described in section 5.

This reduces the amount of load on the network. While recursively applying the aggregation function to these cluster heads in the next level and through multi-hoping through the cumulative information is answered back to the base station.

So an effective routing protocol is necessary to transfer the aggregated information effectively to the base station. In the next subsection we discuss the multi hop routing infrastructure involving in-network aggregation. We study the accomplishment of the task of communication between the sensor nodes of the first level and between the head nodes in the second level, finally transmission from sink node to the base station. Intuitively, data is routed along a reversed multicast tree with the sink as the root. Thus In-network processing can significantly improve the scalability and lifetime of micro sensor networks.

3.4 Routing Model

As a consequence of In-network Aggregation, our work discusses the routing scheme in this section. The reversed multicast tree construction for data-centric routing determined by the following application scenarios of micro sensor networks. Periodic, event driven, query based (which are described in the table below) where the sensors allow all these types of traffic.

Table 3.1 Type of Queries in Sensor Networks

Application Scenario	Description
Periodic	All objects (sensors) report their measurements back to the user once every fixed time interval, as programmed before deployment. All sensors are required to e symmetric.
Event-driven	There will be no data traffic flow from the sensors unless the network encounters or detects some specific events (like movements in military area, or during any habitat monitoring etc.)
Query based	There is a requirement for a routing structure that need to be computed for the query and data transmission between sink and queried sensor(s).

Many previous studies have shown two types of approach that are suitable for routing:

Table 3.2 Routing Schemes

Routing Scheme	Description	Function	Adaptability
Address centric Approach	It uses the logical properties related to the network structure	Each packet is routed based on the unique destination IP address	Does not work with micro - sensor networks
Data Centric Approach	This takes the advantage of data aggregation methodology	The distribution is based on sink broadcasting the query for the data in the network	Well-suitable in the Large/small WSN

Data Aggregation is a technique used to solve many problems concerning Data-Centric routing [16] helps WSN to be perceived as reverse multicast tree [15]. In this thesis, we query model. The queries are intuitively traversed through the computed routes for data transmission between sink node and the rest of sensor nodes.

3.5 Radio Model

Emergence of the concept of distributed wireless sensor networks has gained importance in research and development in the design of short wireless radio range. As a part of communication component it monitors the energy consumption for transmitting and receiving. In our work, we assume the simple First Order Radio Model presented in LEACH [20]. It presumes that the expended energy by the radio for the journey of transmitting and receiving the data bits to be Eelec = 50 nJ/bit, and ε amp =100 J/bit/m2. It also assumes the radio channel to be symmetric, which means the cost of transmitting a message from A to B is same as the cost for transmitting from B to A. It determines the energy loss for channel transmission in terms of r, as r2. Hence, this radio model calculates the energy spent for k-bit packet to send over a distance‘d’ as

$$ETx(k,d) = Eelec * k + \varepsilon\ amp * k * d2$$

$$ERx(k) = Eelec * k$$

ETx is the energy used for transmission, and ERx is the energy used for receiving. It is clear from the above equations, that the transmission energy is almost dependent on the distance parameter. Even though the cost of receiving is relatively low, we have to consider the design of protocols, which can reduce energy cost for both the transmission and receiver circuitry communication.

3.6 Storage Model

Data storage is one of the important resources in the wireless sensor networks. Storage in sensor networks speaks about the space in which data can be saved and the place where query processing can happen. Many previous works have classified storage into following three types.

1) Local storage

2) External storage

3) Data-Centric storage.

Each sensor stores the data on their own storage, the place where they sense, this is called Local storage. Currently the storage is a small amount of memory that is equipped with each sensor. External storage is the data sensed from the sensors is sent to an external storage, it is like providing an outside storage capacity, such as traditional DBMS system to store the collected information. The third one is the data centric storage requires the data are given a specific name based on the type of data produced by sensors and stored at a particular node which is predetermined in the field, for example Geographic Hashing Table GHT[17] . The decision of where to store the data is based on the 'name' rather the node address or position.

Hence the randomly placed sensors with a good storage methodology accomplish the task of reducing the amount of data to be transmitted. It is important to have a suitable storage capability that is adaptable to the kind of sensor filed in use. This is because different kinds of storage techniques allow sensor information to be stored, indexed and accessed efficiently and easily by the applications as required.

In this thesis we consider applications where the sensors sense the data and store it in place (Local Storage) without changing the stored value. This additionally facilitates the primary storage methodology used in our work i.e. the data- centric storage strategy [16].

As mentioned in the earlier section, we assume that the every sensor node can do multi-measurement (e.g. temperature, Humidity, Rainfall, Pressure etc.) For instance,. In the cluster A, node 1 stores only the humidity data. Hence node1 & node 3 send the humidity data to node 2. Whether node1 stores the temperature data or humidity data, depends on the tree structure described in the next section. In-network aggregation with minimum number of transmissions is used for reducing the amount of data to be communicated.

3.7The Storage model in Base station:

In this thesis we consider the base station which is far situated from the sensor field is equipped with GPS. In section V we show the building of routing protocol used in our work. Based on this implementation scheme, the collected meta-data is stored in the base station. The following table lookup reflects the stored information regarding the formation of tree hierarchy on the nodes. And this data is constructed and maintained by the base station, assuming the base station is not energy constrained.

3.7.1 Building the Database

In this section we describe a distributed index structure allows each sensor to efficiently notify if any of the sensors need to participate in a given spatial query. This index structure is built on top of sensor network, which essentially forms a routing tree that is optimized for processing spatial queries in sensor networks.

This index structure imposes a hierarchical structure in the network. We assume the spatial queries will always be disseminated into the sensor network from the base station through the sink node (i.e. a sensor node close to the base station). The base station is responsible for the cycle of preparing the query explicable to the network, submitting it to the sensor field and getting back the result (user compatible). The spatial query is disseminated into the routing tree and the result will be sent back to the root (SINK). The base station having this indexing path in the form of meta-data will decide if the query is applied to the particular node in the cluster or not.

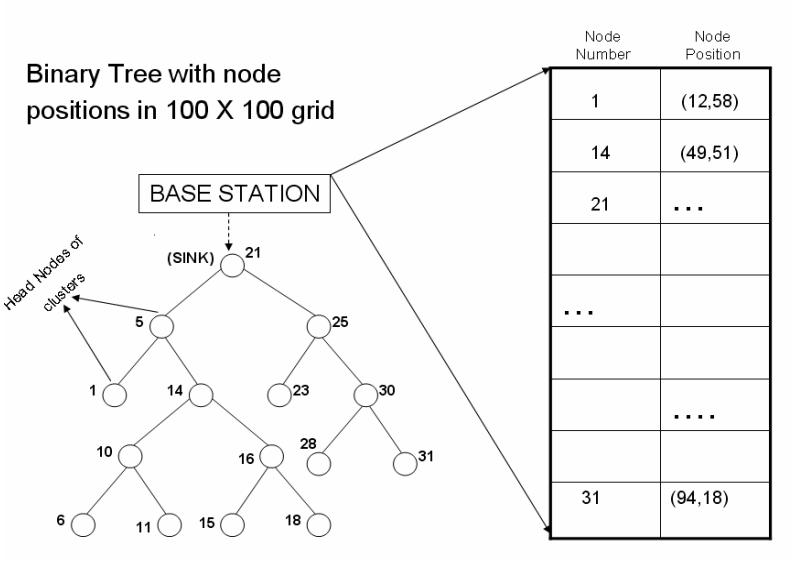

Figure 3.2 Collected Meta-data in Base station

And also the decision is made easy whether the query is applied locally and / or to be submitted to one of its children in the routing tree. A query applied locally if there is a non-zero probability that the sensor produces a result for the query.

Each sensor node in the indexing table maintains a minimum bounding rectangle, this apparently are the clusters formed from the first phase of MULT routing protocol, which is a group of sensors that are close enough to form a cluster and doesn't have to spend energy while in the group. Because, generally we consider the sensors radio range will be the MBR of each corresponding cluster. When a node receives a spatial query, it intersects the query area with its MBR (area). If the intersection is not empty, the base station applies the query and forwards the request to its children. Clearly sensors with smaller MBR area will have higher chance to determine if the query applies to them or to the sensors below them accurately. Therefore, save more energy in processing spatial queries.

This indexing structure exploits the MULT protocol for creating a routing path. The adaptive Binary tree helps in deciding the whole hierarchy of the nodes in the sensor field. Our goal is to minimize the minimum bounding rectangular area and subset of sensor that participate in the answering the query. Thus, increasing more number of small MBR's (apparently clusters) to reduce the overlapping area and consequently reduce the energy consumption in the whole sensor field. This model is proved more effective when the base station query the sensor network based on the distance between the sensor filed and the base station and among the nodes.

Table 3.3 Comparison of the ABT, Quad tree and R tree in the Sensor network

Dimensions	R- Tree	Quad tree	ABT
Load Balancing	Not good	Not good	Highly efficient
Small networks	Depends on the split at nodes	Good	Efficient
Large networks	Depends on the split at nodes	Fair	Highly efficient
Join/Failure	Very difficult to handle	manageable	Handled by the used routing scheme
Query support [Range, Aggregation, conditional….]	Good for Range queries, Nearest –Neighbor queries	Range queries	Aggregation and Conditional queries
Energy efficient	Highly inefficient	Moderate	Highly Efficient
Response time	Good	Fair	Delayed
Indexing area size	Very high	Constant	Less
In-Network Aggregation	Not possible	depends on # of levels	Highly compatible
Redundancy in Indexing	Much higher due to complexity	Much higher	Comparatively less, - suitable for sensors

Table 3.3 Comparison of the ABT, Quad tree and R tree in the Sensor Network

Index creation	Sophisticated	Moderately OK	Simple
Tuning(approximation of Geometrics)	Cannot be tuned	Fine tuned (tiling level)	(Very)Fine tuned (less distance criteria)
Heavy update Activity	Not good choice	Don't effect	Good
Avg. # of sensors per MBR, as increase in # of nodes in 100X100	Depends on split	Constant	Increases
Indexing Dimensionality	4 dimensions	2 dimensions	2 dimensions
Network Topology compatibility	Random	Grid	Grid and Random
Time complexity	Quadratic		Very high
Height balanced	Yes	No	No
Networking support	Broad cast	Broadcast	Hop count

CHAPTER 4

BUILDING ABT & INDEX STRUCTURE

4.1 Construction of Routing Path - An Adaptive Binary tree

In this section we describe the backbone for the index structure which are the MULT algorithms for making clusters and constructing a hierarchy of nodes to aid the indexing structure. First we describe how the sensor nodes are assigned to each cluster used in the clustering process. Second, we present how the clusters are connected into the structure. Finally, we put together all the components in extending it to a decentralized indexing table. We implement two phases in building the tree structure.

4.1.1 Phase 1 of MULT: Creating the clusters

In the Phase-1, we present the clustering algorithm, called MIND, based on the shortest distance from each node. MIND algorithm consists of two parts. The first part is to find the closest node from each sensor node distributed in the sensor field. The second part is to determine which nodes are in the same cluster.

In the first part, all sensor nodes have the program that finds the closest node from each node. Let all sensor nodes set be V={ v1,v2, v3… vn } and all sensor nodes except vi be Vi′= V- {vi}. The closest node defined as Nclosest to vi is defined as follows

$$\text{Nclosest}(vi) = \{vj \in Vi' \mid \min\{d(vj, vi)\}\}$$

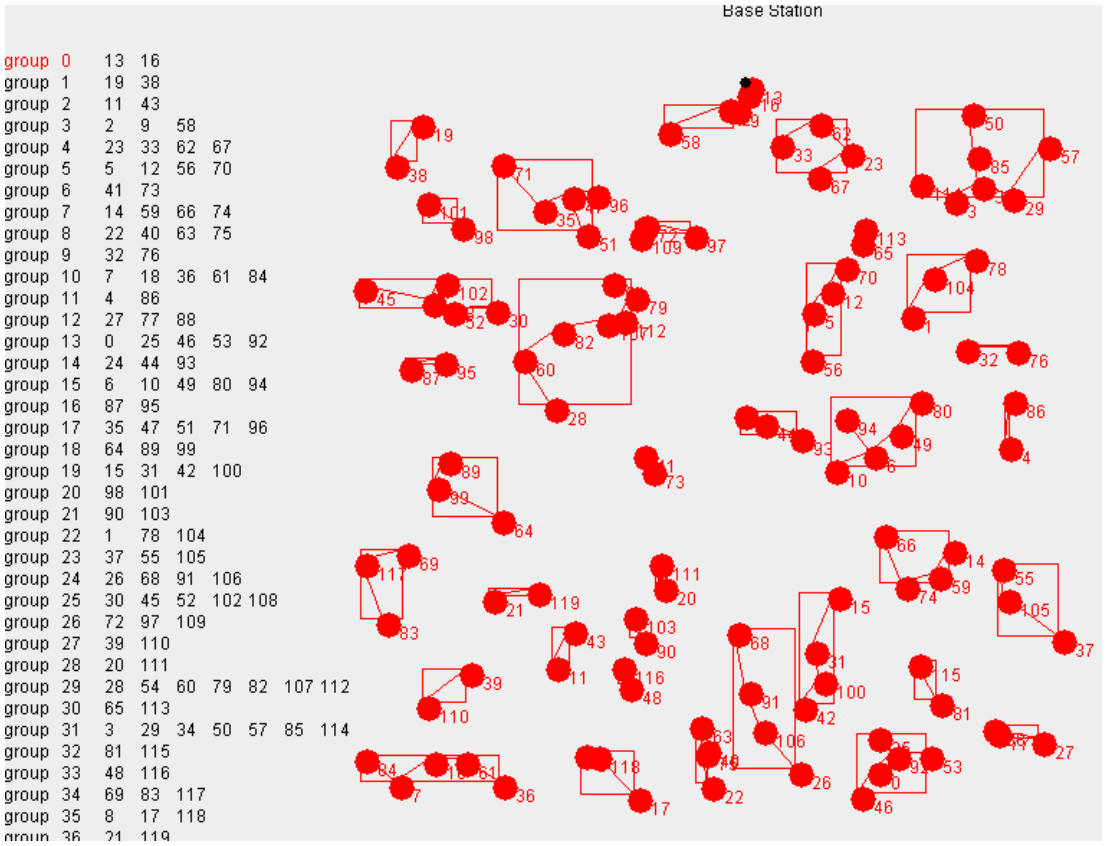

Figure 4.1 Making Clusters

to vi. All nodes can obtain the unique id of the closest node using Algorithm 1. Algorithm 1 works as follows: each node sends its position and node id to all nodes within its radio range. Each node then calculates from the received information its closest node. For example, in Fig. 9, node 3 and node 1 are in the radio range of node 2. The node 6 is out of radio range of node 2. Therefore node 2 received {id, position} from nodes 1 and 3, and calculates node 1 as its closes node. Similarly, node 3 is the closest node to node 2. Hence node 2 calculates the result of algorithm 1 as {id of node 1, position of node 1}. All nodes send the output of algorithm 1 to the one node among the neighbor node in the radio range that is the closest node to the base station. Repeatedly, this neighbor node sends the information received from previous node to another neighbor node which is the

closest node to base station. In this way, the base station obtains the output of algorithm 1 from the all sensor nodes through multi-hop routing.

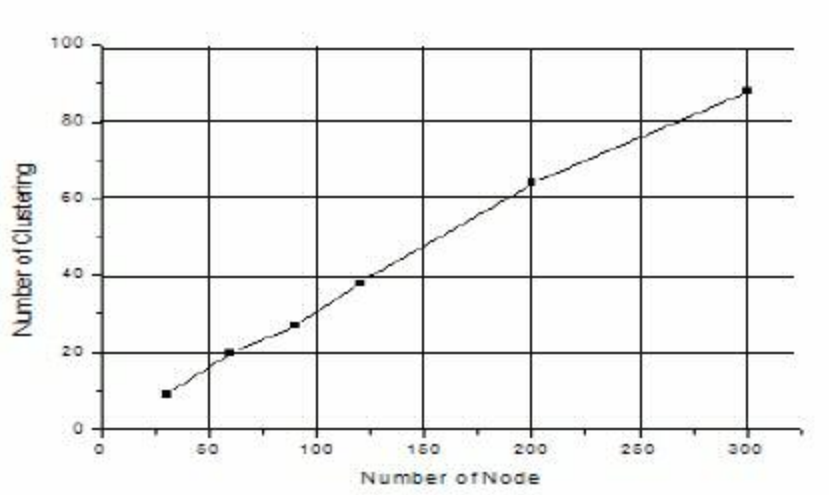

Figure 4.2 showing the increase in number of clusters as the increase in number of nodes

In the worst case of clustering, we consider that all sensor nodes form one gigantic cluster. However as the number of nodes is increased, number of clusters is also increased. Fig.10 is a graph that illustrates the result of the number of cluster. In the second part, the base station which has received the result of Algorithm 1 from all sensor nodes creates the Table 1. Table 1 is the input data for algorithm 2 which is used to determine which nodes are in the same cluster. In algorithm 2, variable i is the id of each node and variable j is the id of the closest node to each node. The result of Algorithm 2 is Table II. In Table II, node[1], node[2] and node[3] have same value as 3, also, node[4], node[5], node[6] and node[7] have same value as 7. This result implies node 1, 2, and 3 are in the same cluster and node 4, 5, 6 and 7 are in the same cluster. Through the Phase-1, all the randomly placed sensor nodes in wireless sensor network form into self-organized clusters.

4.1.2 Phase 2 of MULT: Connecting the clusters

In Phase-2, we illustrate how to decide the head node in each cluster and connect each head node of cluster into a tree structure. After the base station builds all tree structures using the data received from all sensor nodes, base station sends the information necessary to build routing tree to all sensor nodes. As mentioned earlier in the in-network aggregation, the node which is the closest node from base station become a first level root node of one tree structure of the sensor field and at the same time, also becomes the head node of a cluster.

The first level root node is defined as Rlevel 0. In the next step, base station selects the closest node defined as R1level 1 and second closest node defined as R2level 1 from Rlevel 0 using the data sent from all sensor node. R1level 1 and R2level 1 are chosen to be in the different clusters than Rlevel 0. Also R1level 1 and R2level 1 are not in the same cluster and become head nodes of each cluster. After constructing the first tree, the base station sends the information describing the tree structure to the node Rlevel 0. Also Rlevel 0 stores which nodes are in the same cluster with Rlevel 0 and which nodes are R1level 1 and R2level 1. Then, Rlevel 0 sends the information describing the tree structure to R1level 1 and R2level 1. In this way, the tree routing information is transferred to the last level cluster head node. Fig. 11 shows the ABT. Algorithm 3 shows the way to build the tree structure. The array distance[i] has the distance value from the head node of each cluster to all other nodes. The array cluster[i][j]=z is that variable i has the identification number of a cluster and j is the node number of a node in the same cluster and z is the identification number of the node. For example, cluster[3][2]=4 means

that node 4 belongs to cluster 3 and is ordered as second node in cluster 3. In line 19 of Algorithm 3, if variable new has

0, this cluster was not connected to the tree. Therefore this cluster has to be connected to the tree and we update variable new to be 1. If variable new has 1, this cluster was already connected to the tree. Therefore this cluster is skipped.

A sample adaptive Binary tree for the sensor network with 120 sensors randomly distributed in a sensor field of size 100 X 100 is shown below.

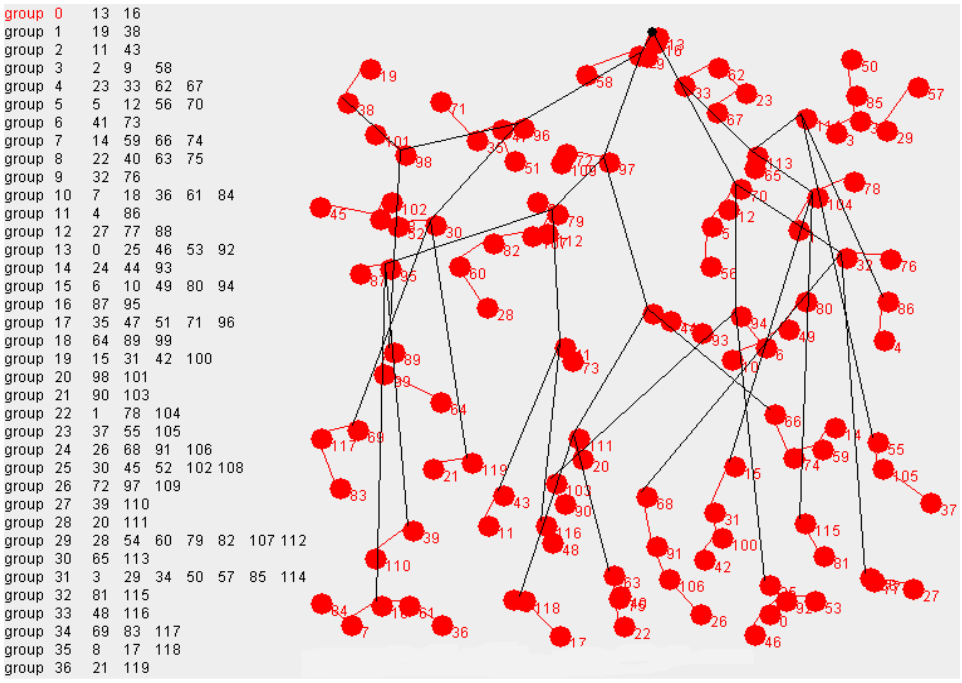

Figure 4.3 Making of ABT

4.2 Construction of the Indexing Structure with ABT

So as and when the tree structure is formed, the information is calculated along with the complete family history of each node such as its position, ,its parent, child, MBR coverage area, in the base station as shown in the Table 2. Thus, these calculations lend a hand in making the decision on which path has to be chosen on the

routing path to better disseminate the spatial query efficiently. So, the in-network computations are performed only on subset of nodes, reducing the number of nodes that participate in answering the query which markedly effects low consumption of energy. Thus the table previously shows the way in which the meta-data is stored in the base station to resolve the spatial queries. For example, calculating the AVG temperature in area 'X'.

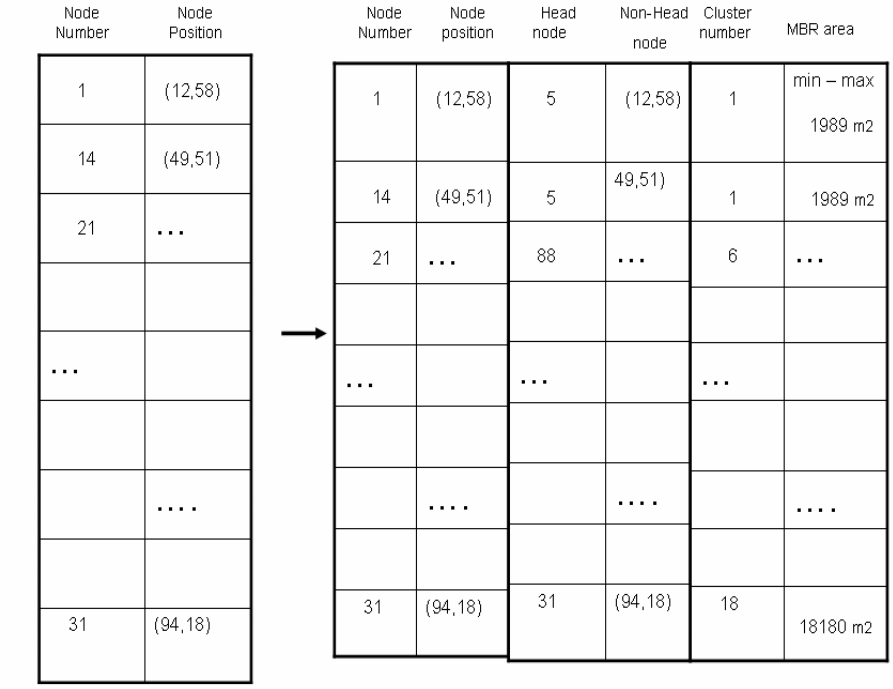

Figure 4.4 Index construction using the metadata in base station

From the below Fig 10 the total area has five MBR's intersecting the query window of area X. It is clear from the stored information that A, C, B, D are the head nodes participating in the path as well as parents for the other nodes from other MBR i.e. the nodes of some other cluster heads which are not participated in area X. But from the look up table, the parent of any node is known and the query is also decided to be traced into the path of head node exterior to the query region.

Thus it is quite simple to know the number of times the messages has to be processed and area in which the query has to be circulated is condensed to a great extent. Apparently, only few subset of nodes which are selected in a are pre-determined way with the criteria of intersecting the query region are likely to participate in receiving signals and answering the query. Analyzing the number of times the query to be processed per area and the total number of sensors participating in expending energy is shown in the next section of our evaluation

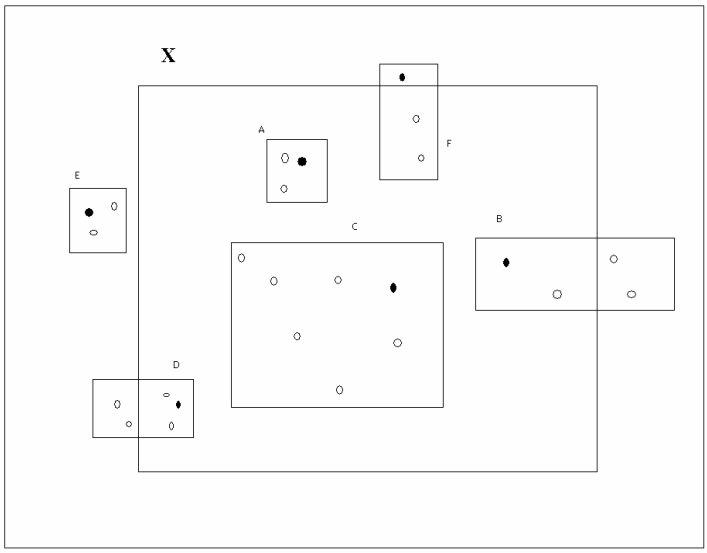

Figure 4.5 Number of nodes in the MBR that intersect the Query MBR

4.2.1 Maintenance and optimizing phase

During the maintenance phase and optimizing phase, a sensor node might join or leave the network. This may happen in the following conditions:

1. One or more sensors are added to the network
2. A sensor fail to respond and thus it must be removed from the indexing structure

3. A sensor did not join the network during the building process.

4.2.1.1 Sensor Join/Fail

In the sensor network we assume that sensors are static and are not aware of their locations .We need to consider the situation where the sensors may join or fail from the network anytime. Spatial queries are always submitted from the base station and the query result is sent back to the base station. We also regard sensors to be homogeneous in transmitting and processing power, using the same energy in processing a task and submitting a radio signal. As in cougar [23], TAG [6] and TINY DB [5], each sensor maintains data as a single table with two columns for the sensor X and Y co-ordinates. Queries are parsed and disseminated into the network at the base station. The Spatial query over the sensor network can return a set of information or an aggregated data in a particular area 'X' of the sensor network.

The whole MULT algorithms adjust to the total number of sensors in the field. When a sensor determined that it needs to connect to the network it runs Algorithm 1 and Algorithm 2. This results a new set of clusters and new subset of head nodes with transformed MBR's. Thus each time this algorithm runs the changes in the tree hierarchy have to be done all the way back to indexing structure in the base station.

Sensors may fail at any time and each sensor is forced to re-compute using the Algorithm 1 of phase1. Every time the base station should keep track of all the nodes and the list of children, the parent child relationship, recalculating the area. Not even a node or any section of nodes does not become orphans. Even if a single node in the

field is left unconnected (due to limited radio range), it is eventually made as a head node and connected to the network. In this way the network is never disconnected.

CHAPTER 5

IMPLEMENTATION AND EVALUATION

5.1 Implementation

We have implemented the indexing modules and the simulated program in Java. We used our implementation to experimentally evaluate our technique.

5.2 Evaluation

The goal is to show that our scheme provides an efficient distributed way to execute spatial queries in the sensor network. Conceptually, our method is a decentralized spatial index over the sensor network that can be used to locate the sensor nodes which have relevant data to the spatial query.

From the indexing table, all nodes that intersect the query range are calculated by means of stored MBR's. Initially, we get the head nodes and check all the nodes which are related to that cluster, if they intersect with the query window. Then all the remaining nodes which doesn't fall under this MBR whose head node is not in the query region. Here again from the look up table the nodes are checked and the query is just forwarded to the head node and executed by the non head nodes that fall under the query window and participate in answering the query. If the node MBR does not intersect the queried region, the query will not be forwarded. Also if the query applies locally, it executed the query. If the query does not apply to the sensor s and its nodes inside the rectangle, it is dropped.

We can differentiate two sets of sensors that participate in the query.

A. Implementer: The set of sensors that the query applies to them
B. Dispatcher: These are the set of sensors where the query is just allowed to be forwarded to the implementer.

An implementer sensor can also be a Dispatcher, and forward the query to the other sensor. The number of implementer in a spatial query is a function of the query area and the distribution of the sensors. The number of Dispatcher sensors depends on the way that we route the queries towards the implementer. For each sensor network, we built a index using the routing tree. Each sensor chooses geographically closest sensor and form the clusters. The head node selection criteria is described in the section 4.1.2 to study the effect of MBR formation and the coverage area of each cluster in the processing spatial queries.

5.2.1 Evaluation parameters

We used the following parameters to evaluate the indexing structure with the traditional methods like R-tree and Quad tree.

I. Energy Consumption
II. Number of sensors involved
III. Indexing Area covered

5.2.1.1 Number of sensors involved

This describes the number of sensors that are involved in the index that intersect with the query MBR. According to the indexing structure in the base station, all the

nodes information such as parent, child, Head, non-Head are calculated. From this we can look up all the nodes MBR's that intersect with the query MBR. We have classified the participation nodes into three categories

1) Sensor, if it is head node in the query region, the query is executed as well as forwarded for the nodes which are having the same MBR.
2) Sensor, if it is a non-head node intersecting the query region also performs the same action as the head node intersecting the query region
3) Sensor which are either head/ non-head node non intersecting the query window
    a. If it is a head node of the MBR intersecting the query region, it just forwards the query.
    b. If it is non-head node then it is neglected from participating in answering the query.

5.2.1.1.1 Average number of sensors that participate in spatial query

Figures 5.1- 5.4 shows that number of sensors in the MBR's of the index that intersect with the query MBR. And the nodes over variable size of the query areas are shown. The X-axis shows the query area size to study area size. The Y-axis shows the number of participating nodes. Each point in the graph is obtained by averaging 50 randomly constructed spatial queries. For each different query area size, we try to evaluate our scheme with R-tree and Quad tree. As per our experiments, those sensors MBR that intersect with the querying MBR is more for

R-tree and Quad tree. Our study show that as the area decreases the number of sensors participation eventually decreases. This helps in conservation of energy.

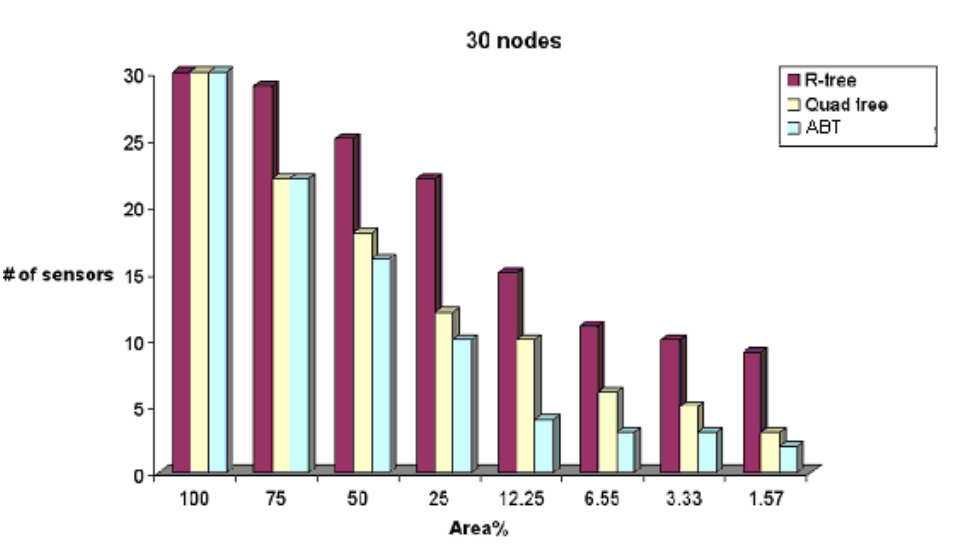

Figure 5.1 Number of nodes calculation with 30 nodes

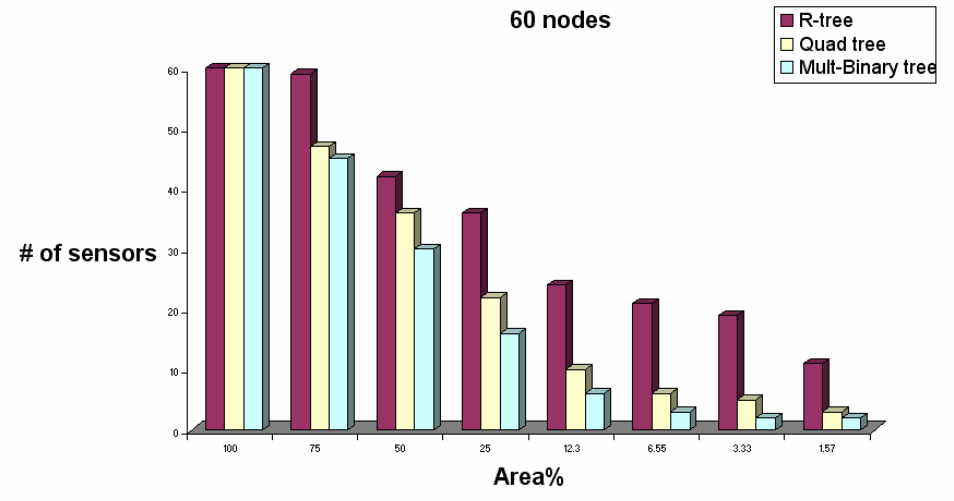

Figure 5.2 Number of nodes calculation with 60 nodes

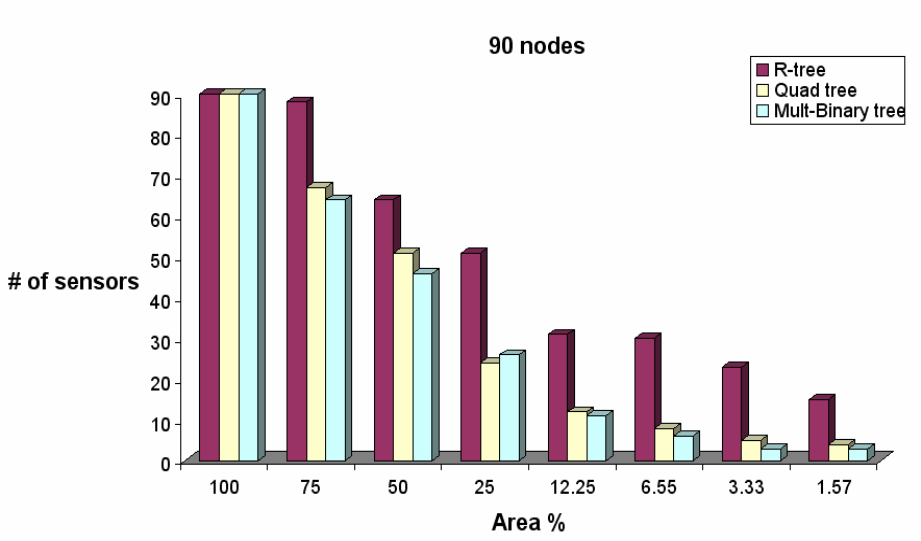

Figure 5.3 Number of nodes calculation with 90 nodes

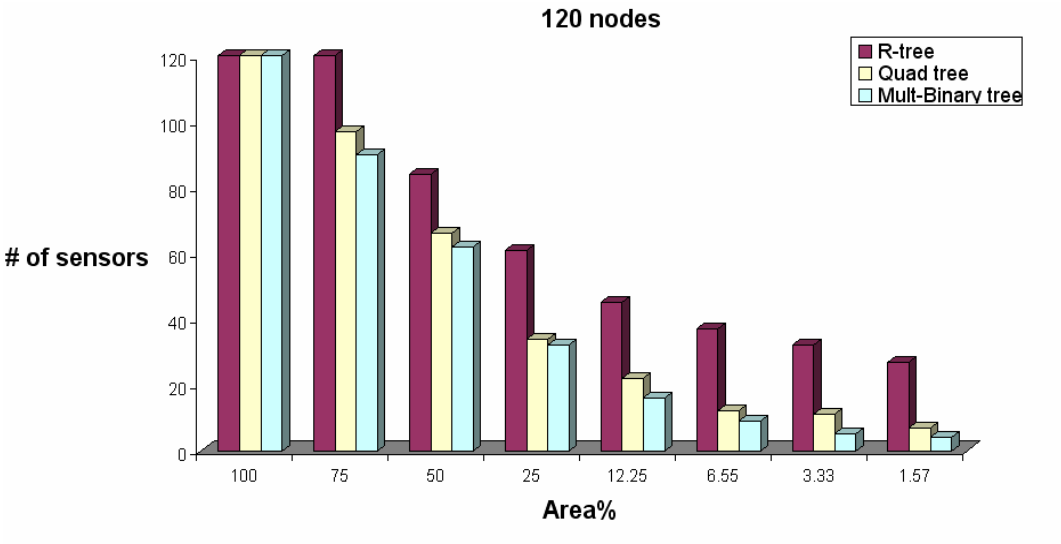

Figure 5.4 Number of nodes calculation with 120 nodes

We made to run the query with change in number of nodes from 30, 60, 90 and 120 with the change in query area size. In query area size of 100 X 100, shows that our work outperforms the traditional R-tree and Quad tree.

5.2.1.2 Energy Consumption

To evaluate the power consumption we used the radio model of LEACH[20] for energy consumption in sensor nodes. Each point in these graphs is obtained by averaging 50 randomly constructed spatial queries. We assume that the energy required for communications O (d^2) where d is the Euclidian distance between the two nodes.

Figures (5.5 – 5.9) show that the power consumption in processing the spatial query in sparse networks through the dense (i.e. 30 – 120 nodes) in 100 X 100 grid. The X-axis represents percentage of ratio of query Area to the study area. The Y-axis is Energy consumption in Joules. Our experiments prove that when there is increase in number of nodes (i.e. density), the distance between each node decreases. Eventually, as the number of MBR's increases the energy consumption per area decreases. But this increases the number of times the query to be disseminated to the network for it be processed. And as the number of nodes increases, the total energy consumption increases slowly. If there is increase in number of nodes, the distance between each node will be very less and therefore the total energy consumption is very slow growing function of distance. In the result, our indexing depicts better results than others.

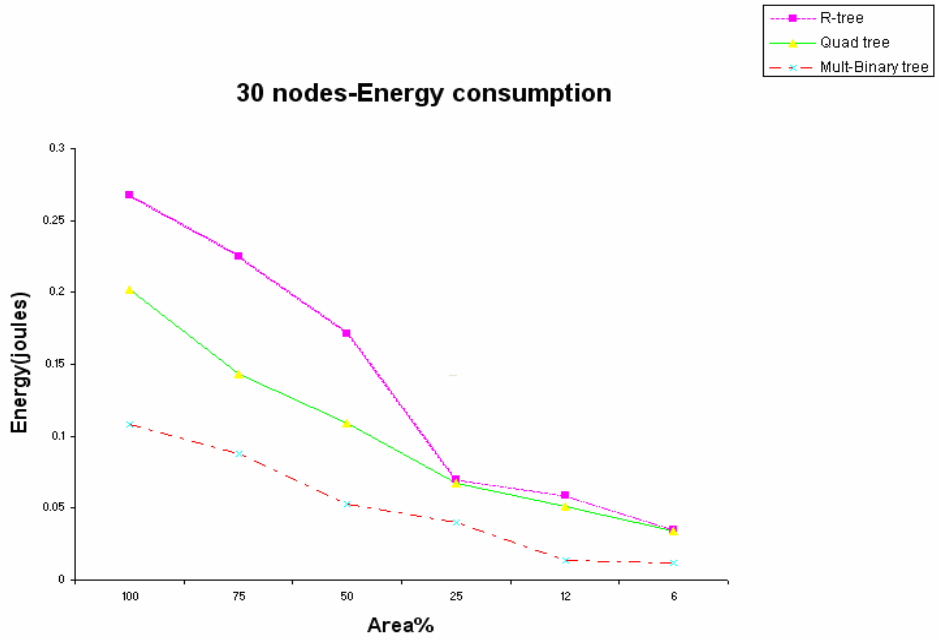

Figure 5.5 Total Energy calculations for processing query with 30 nodes

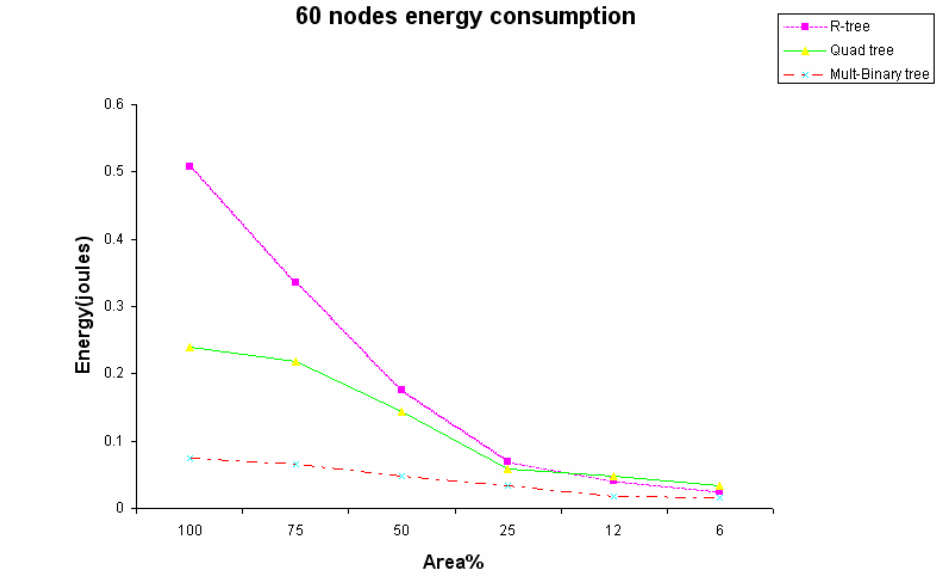

Figure 5.6 Total Energy calculations for processing query with 60 nodes

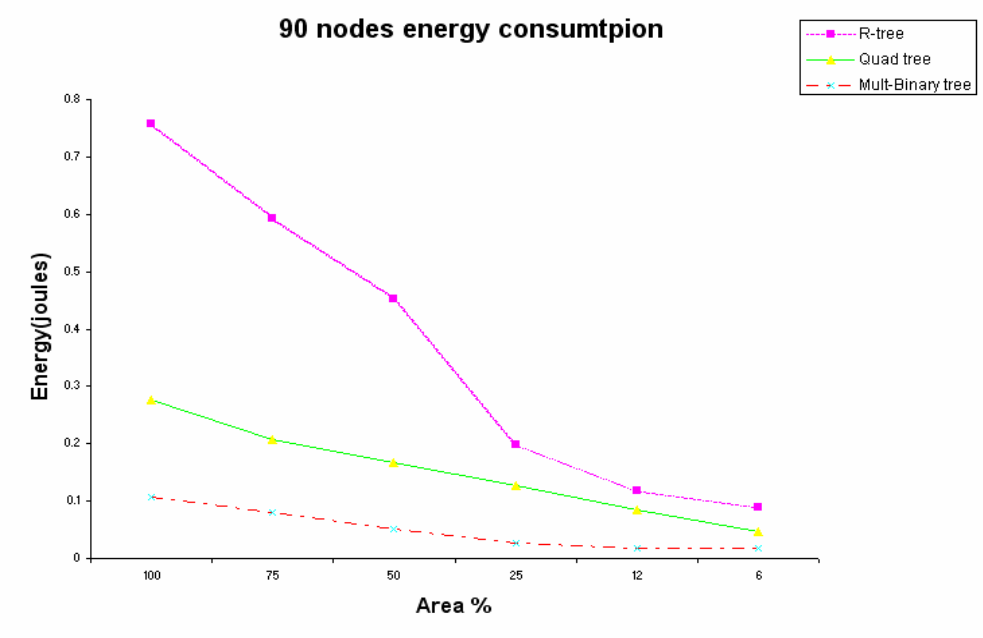

Figure 5.7 Total Energy calculations for processing query with 30 nodes

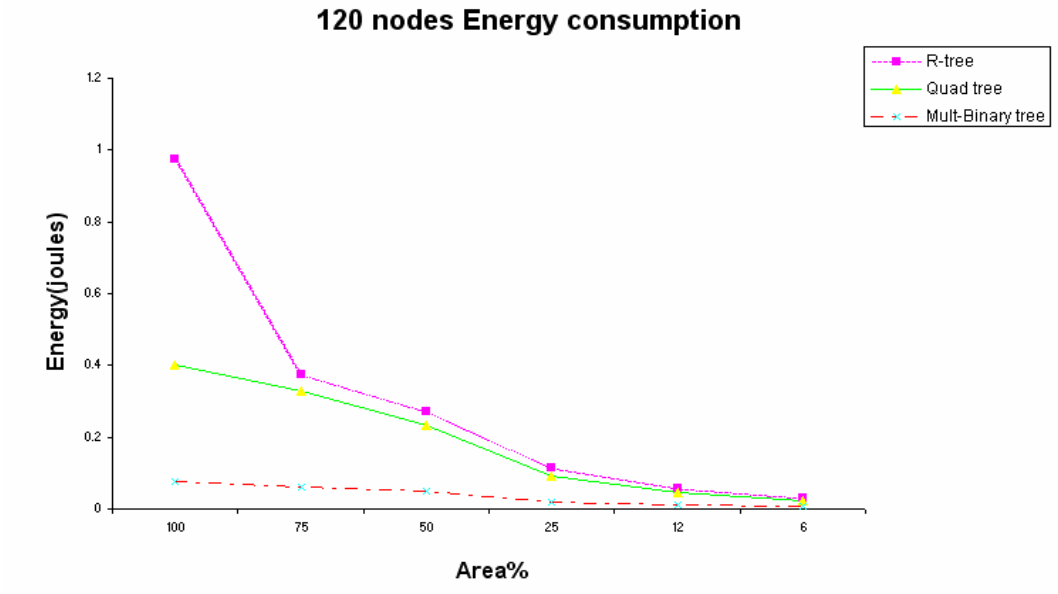

Figure 5.8 Total Energy calculations for processing query with 30 nodes

5.2.1.3 Indexing Vs the Field area

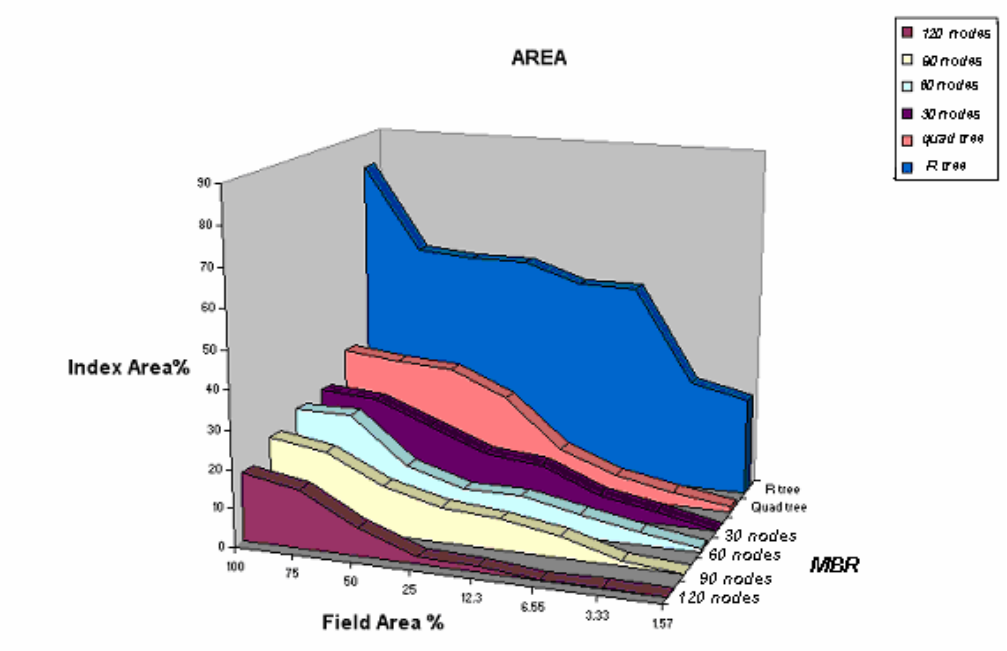

Figure 5.9 Indexing Vs Field Area

As in the fig 5.9, it can be seen that area that need to be queried intersects the MBR area that need to be queried intersects the MBR area for each of the indexing structures , R-tree , Quad tree and our MULT-adaptive binary tree. As per our study and results R- tree requires 3 X area required for the query to be processed by ABT. Also, the results and the graphs prove that as the number of nodes increases the area size to be queried also decreased in slow manner. This clearly affects the amount of energy consumed.

CHAPTER 6

SUMMARY OF EXPERIMENTAL RESULTS

Our experiments show that as the density increases, ABT produces a tree , where the majority of rectangles have small geographical size. Consequently, ABT is very efficient in evaluating spatial quires. The simulation results show that in sparse or dense networks – the decision of subset of nodes with small % of overlapping will participate in spatial query processing, when the spatial query window is a rectangle. Therefore, it would be more energy efficient to use this criterion in sensor networks .The experimental results show that the power consumption and the number of sensors in MBR's of index that intersects the query MBR in ABT is lower than the other models evaluated.

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this thesis, we presented ABT, a decentralized index for distributed spatial query execution in sensor network. ABT provides an efficient way to access sensor data in distributed manner while preserving energy and reducing the communication cost. ABT include a Maintenance and Optimization phase, which enables the network to reconstruct itself when the sensor fails or allows the sensor to join splitting the cluster into smaller ones, which minimize the MBR area, increasing the number of MBR's. This mechanism will refine the structure of the index to respond to the failure and save energy in spatial query processing.

We have evaluated the behavior of the ABT in sparse and dense networks and show that this spatial index provides an effective method to run spatial queries. In particular, we studied the effect of making the clusters and choosing the head nodes based on the distance. Using the simulations, we got the ABT experimental results of deciding the set of nodes who's MBR intersect with the query window and are reached with the help of Binary routing path constructed. The simulations show that making MBR's with nodes implementing multi-hop in the Cartesian co-ordinates will involve fewer sensor in processing a spatial query. Thus, eliminates the nodes which does not satisfy the spatial constraint and therefore saves energy.

We have identified several opportunities for future research. ABT is designed in such a way that it can respond only when the query is submitted from the base station to the sensor field which is Cartesian co-ordinate system. We can extend the indexing structure to polar co-ordinate system. This can be done submitting the spatial query to the polar coordinates. We have evaluated the aggregation queries. Also our work can be extended to other kinds of queries such as nearest neighbor or range queries.

APPENDIX A

ALGORITHMS

Algorithm 1 : Finding the closest node

Input to each node: position and unique id information of all nodes in the radio range of each node.

1. Compare distance to all nodes in the radio range.

2. Find Nclosest(vi) = {vj¸Vi Œ | min{d(vj, vi)}.

Output: unique id of the Nclosest(vi) .

Algorithm 2 : Finding the same cluster

Input: variable i = node id , variable j =id of the closest node.

1: Set all arrary node[i] = NULL

2: for(i=1 ; i<= number of node ; i++)

3: if (node[i] != NULL)

4: tmp = node[i]

5: for(k=1 ; k <= number of node ; k++)

6: if(node[k] == tmp)

7: node[k] = i

8: if (node[j] != NULL)

9: tmp = node[j]

10: for(k=1;k<=number of node ; k++)

11: if (node[k] = temp)

12: node[k] = i

13: if (node[j] == NULL)

14: node[j] = i

15: if (node[i] == NULL)

16: node[i] = i

Algorithm 3 : Building the tree

variable i is id of each sensor node.

set c=0 , number of branch =0

1: do{

2: for (i=0 ; i<number of node ; i++)

3: distance[i] = distance(head node , i)

4: for (i=0 ; i< number of node ; i++)

5: sort_distance[i] = sort by ascending(distance[i])

6: for (i=0 ; i<number of node ; i++)

7: for (j=0;j<number of node ;j++)

8: if (cluster[i][j]= head node)

9: i is cluster number with head node

10: for (i=1 ; i<number of node ; i++)

11: for (j=0 ; j<number of node ; j++)

12: if (sord_distance[i]==distance[j])

13: for (k=0 ; k<number of node ; k++)

14: for (m=0 ; m< number of node ; m++)

15: if(cluster[k][m] == j)

16: k is the closest node from the head node

17: for (n=0 ; n<number of node ; n++)

18: if (k != used_cluster[n])

19: new =0

20: else

21: new = 1

22: if (new=0)

23: used_cluster[c]=k

24: c=c+1

25: connect(head node, i)

26: branch number = branch number +1

27: if (branch number ==2)

28: break;

29: } while(end of node)

Algorithm 4:

R-tree: Algorithm Quadratic Split Divide a set of

M+1 : index entriees into two groups

Qsl

[Pick first entry for each groups]

Apply Algorithm PickSeeds to choose two entries to be the first elements

of the groups

Assign each to a group

Qs2

[Check If done]

If all entries have been assigned, stop. If ,one group has so few entries that all the rest must be assigned to it m order for it to have the minimum number m, assign them and stop

[Select entry to assign]

Invoke Algorithm PickNext to choose the next entry to assign Add it to the group whose coverage rectangle pull have to be enlarged least to accommodate it Resolve ties by add mg the entry to the group smaller area, then to the one fewer entries, then to either Repeat from QS2

Algorithm Pick Seeds

Select two entrees to be the first elements of the groups

PSl [Calculate inefficiency of grouping entries together] For each pair of entl-les El and E2, compose a rectangle includmg El I and E2 I Calculate d = area(J) - rea(El I) - area(E2 I)

PS2 [Choose the most wasteful pair]

Choose the pair with the largest d

Algorithm Pick Next Select one remaining entry for classification in a group.

PNl

[Determine cost of putting each entry m each group]

For each entry E not yet m a group, calculate d,= the area increase required m the covering rectangle of Group 1 to include EI Calculate d2 similarly

For Group 2

PN2

[Find entry mth greatest preference for one group]

Choose any entry with the maximum difference between d 1 and d2

REFERENCES

[1] Ossama Younis, Sonia Fahmy, "Distributed Clustering in Ad-hoc Sensor Networks : A Hybrid, Energy-Efficient Approach," IEEE Transaction on Mobile Computing Vol.3, No.4, Oct-Dec 2004

[2] A. Gutman, RTree – A dynamic index structure for spatial searching, SIGMOD 1984, Boston, MA, 1984.

[3] N. Beckman, H. Kriegel, R. Schneider and B. Seeger. The R*tree: An efficient and robust access method for point and rectangles. SIGMOD 1990.

[4] T. K. Sellis, N.Roussopoulos and C. Faloutsos. The R+ tree: A dynamic index for multi-dimensional objects. VLDB 1987, Brighton, England, Sept 1987.

[5] H. Hu, J. Xu, W.S. Wang, B. Zheng, D. L. Lee, W Lee, Proactive Caching for Spatial Queries in Mobile Environments,

[6]S. Madden, M. J. Franklin, J. M. Hellerstein and W.Hong, A Tiny AGgregation service for ad-hoc sensor Networks, OSDI, 2002, Boston, MA.

[7] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann and F. Silva, Directed Diffusion for Wireless Sensor Networking. IEEE/ACM Feb. 2003.

[8] D. A. Coffin, D. J. Van Hook, S. M. McGarry, and S. R. Kolek. Declarative ad hoc sensor networking. SPIE Integrated Command Environments Conf., San Diego, CA 2000.

[9] Y.-B Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. Mobicom'98, Dallas, TX, Oct 1998.

[10] O. D. Sahin, A. Gupta, D. Agrawal and A. El Abbadi, A Peerto- Peer framework for caching range queries. ICDE 2004, Boston, MA, 2004.

[11] B. Zheng, J. Xu, W. C. Lee, and D. L. Lee: Grid-Partition Index: A Hybrid Approach to Nearest-Neighbor Queries in Wireless Location-Based Services, VLDB Journal, 2004.

[12] A. Gupta, D. Agrawal and A. El Abbadi. Approximate range selection queries in peer-to-peer systems. CIDR 2003, Asilomar, CA.

[13] M. Ettus. System Capacity, Latency, and Power Consumption in Multihop-routed SS-CDMA Wireless Networks. In Radio and Wireless Conference (RAWCON '98), pages 55– 58, Aug. 1998.

[14] Bhaskar K., Deborah Estrin, S.,Wicker "The Impact of Data Aggregation in Wireless Sensor Network," The work supported by DARPA SensIT program. Cornel Univsersity.

[15] Ian F.Akyildiz ,Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirici, "A Survey on Sensor Networks," Georgia Institute of Technology.,IEEE communication Magazine ,August-2002.

[16] Shenker,S.,Ratnasamy,S.Karp,B.Govindan,R., and Estrin,D,".Data Centric Storgein Sensornets," To appear in the First ACM SIGCOMM WorkShop on the HotTopics in he Networks (HotNets 2002),Princeton NJ, October, 2002.

[17] K. Park, R. Elmasri, "Effects Of storage architecture on performance of sensor network queries," In ICOIN 2006

[18] B.Y. Lee, Raja R Anugula, Dr. Ramez Elmasri "Energy Efficient in-Network Aggregation Using Multiple Trees in Wireless Sensor Networks" , UTA,Arlington,Tx, March, 2006.

[20] Chandrakasan, Amirtharajah, Cho, Goodman, Konduri, Kulik, Rabiner, and Wang. Design considerations for Distributed Microsensor Systems. In IEEE 1999 Custom Integrated Circuits Conference (CICC), pages 279–286, May 1999.

[21]Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan "Energy-Efficient Communication Protocol forWireless Microsensor Networks**"** Massachusetts Institute of Technology Cambridge, MA 02139

[23] P. Bonnet, J. Gehrke and P. Seshardi, Toward sensor database systems, Conference on Mobile Data Management, 2001.

[24] S. Madden, M. J. Franklin, J. M. Hellerstein and W. Hong, The Design of an Acquisitional Query Processor for Sensor Networks, SIGMOD 2003, San Diego, CA.

BIOGRAPHICAL INFORMATION

Raja Rajeshwari Anugula was born in Hyderabad, INDIA. She has earned her Bachelor's degree in Computer Science and Information Technology, in Jawaharlal Nehru Technological University, Hyderabad, India, in May 2004. She joined NJIT for Masters Degree and after completing first semester she transferred UTA in January 2006 to continue the studies and pursue the Masters Degree in Computer Science and Engineering. Her major areas of interest are Database System, Networks and Advanced Databases. She earned her asters in Computer Science degree in August 2006.

.