ARCHITECTURES AND METHODS FOR ENERGY-EFFICIENT QUERYING AND INDEXING IN WIRELESS SENSOR NETWORKS

by

KYUNGSEO PARK

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

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2008

To my wife Jihye and my daughter Seyon.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Ramez Elmasri for constantly motivating and encouraging me, and also for his invaluable advice during the course of my doctoral studies. I wish to thank my academic advisors Dr. Hua-Mei Chen, Dr. Leonidas Fegaras, Dr. Yonghe Liu, and Dr. Roger Walker for their interest in my research and for taking time to serve in my dissertation committee.

I would also like to extend my appreciation to chair of the Department of Computer Science and Engineering, Dr. Fillia Makedon and Dr. Bob Weems for their giving me a teaching assistantship during my doctoral studies. I wish to thank Hunsuk Lee, who has encouraged me and has given me religious faith whenever I struggled with my uncertain future. I am especially grateful to Dr. Jeongkyu Lee for giving me valuable advice for my entire work. I wish to thank Byoungyong Lee, Jaesung Choi, and Hyun Lee for frequent academical discussion of my works. I wish also to thank Brittany Lee and Tony Lee for their proof reading of work that I have done.

Last but not least, I would like to express my deep gratitude to my parents and parents-in-law who have inspired me and sponsored my graduate studies. I am also extremely grateful to my wife, Jihye and my daughter, Seyon for their sacrifice, encouragement and patience. I also thank several of my friends who have helped me throughout my career.

June 16, 2008

ABSTRACT

ARCHITECTURES AND METHODS FOR ENERGY-EFFICIENT QUERYING AND INDEXING IN WIRELESS SENSOR NETWORKS

Kyungseo Park, Ph.D.

The University of Texas at Arlington, 2008

Supervising Professor: Ramez Elmasri

Wireless sensor networks have been an active research area for about a decade due to their adaptability to applications involving long-term environmental monitoring. Recent technologies make it possible that a small device equipped with multi-purposed sensors, small CPU and memory, wireless transmitter and receiver, and software can form a network, measure some quantitative phenomena and communicate with each other seamlessly. In wireless sensor networks, one of the basic applications is to monitor events and measure values at places that people cannot reach easily or where a long term sensing task is required. In this case, we need to know the properties of diverse types of sensor network queries and data that are different from traditional ones. Also, we need to solve the intrinsic sensor network problem, energy saving, since users want to gather data for a long term and it is generally not feasible to replace batteries in sensor devices after the sensors are deployed.

In order to solve this problem, first, we classify sensor network queries and find a suitable network system that includes different routing and storage types for each type of query. Second, energy efficient indexing and data gathering methods for sensor networks are studied. In energy efficient indexing, we propose a sectioned tree index, which divides the network area into

several squares and each square has a local index subtree organized within that square. Local trees are interconnected to form one big tree in the network. Local trees are also built based on any algorithm that is energy consumption aware at each sub-root node in a locally centralized way. In energy efficient data gathering, we create energy-efficient data gathering routing tree when we consider two-way communication for reliable transmission and collision prevention. This makes the problem intractable, and we prove our problem is NP-complete by showing that a known NP-complete problem is a special case of our problem. In order to get an energy-efficient routing tree, we propose several heuristic techniques that backtrack one or two steps (BT1 and BT2). We give various values as parameters in measuring energy equation and simulate our proposed algorithms in diverse conditions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Sensors are devices that sense or detect natural phenomena, such as temperature or humidity. Many sensors have now become multi-functional devices that incorporate advanced micro electronics and wireless communication technology. Sensors have become multi functional devices that not only sense multiple phenomena but also perform multiple functions other than sensing or measuring [2, 3]. A sensor platform can be composed of four units: sensing unit, processing unit, communication unit, and power unit [2]. This device can be as small as a penny and still contain multi functional sensor for sensing unit, small CPU and storage device such as memory for processing unit, wireless transmitter and receiver for communication unit, and batteries for power unit.

These sensors can be used to measure some physical values, detect phenomena, or monitor a certain area to determine whether an event occurs or not. Often these sensors are deployed in areas where people cannot reach easily or where a long term sensing task is required. Hundreds or even thousands of these sensors form networks so that they can measure, detect, monitor phenomena and transfer their measured data wirelessly to a certain location where users such as scientists or researchers can analyze them. This technology is known as wireless sensor networks (WSN). A typical scenario of WSN is as follows: Sensors are randomly deployed at an area of interest. After a certain amount of time, they communicate with each other to know where their neighbors are. Then they build their own network so that their measured data can be transferred to a base station by multi-hop routing. Queries can be submitted by users later. The queries can be disseminated through the network and the sensors perform the

tasks required by the query. The sensors can then report data to the base station depending on the queries they received.

Based on the characteristics mentioned above, wireless sensor networks are used in many areas. Examples of applications include military or surveillance applications [4, 5, 6], habitat monitoring [7, 8, 9], environmental monitoring such as volcano, wildfire, and water contamination [10, 11, 12, 13], structural monitoring such as bridges, mines, and buildings [14, 15, 16], underwater monitoring [17, 18, 19], healthcare and medical care monitoring [20, 21, 22, 23], and traffic and parking monitoring [24, 25, 26].

The technologies of wireless sensor networks are mainly based on the technologies developed for wireless ad-hoc networks. In wireless ad-hoc networks, data are forwarded via nodes and the next hop can be determined dynamically depending on network connectivity because of node mobility. Wireless sensor networks have been developed on the properties of wireless ad-hoc networks such that the networks communicate using multi-hop techniques, need multi-hop routing protocol, and communicate with each other seamlessly. However, there exist differences between traditional wireless ad hoc networks and wireless sensor networks [2]:

- The number of nodes in sensor networks can be much larger than in ad-hoc networks.
- The density of node deployment in sensor networks can be higher than in ad-hoc networks.
- Sensor nodes are more likely to fail for communication.
- Sensor nodes are more limited in energy, memory capacity, and computational capacity.

There are also research challenges in wireless sensor networks, such as the following:

- Energy management due to limited battery power and long term monitoring.
- Topology control, deployment
- Localization, time synchronization
- Scalability

- Coverage
- Route discovery and maintenance
- Efficient data gathering, aggregation, and in-network processing
- Efficient processing of massive amounts of data.
- Data storage
- Security, privacy, resiliency, and fault-tolerance
- Real time data acquisition and analysis
- Quality of services

The main research contributions of this thesis are in analyzing and developing energy efficient indexing and data gathering techniques for wireless sensor networks. As a prelude to this research, we first overview and evaluate the different types of sensor networks and different types of queries that are used in WSN applications. There are many kinds of schemes, algorithms, structures, and architectures for WSN to solve a particular problem. Different schemes have their own characteristics that fit best in particular situations. Much of the previous research has focused on individual algorithms in each branch of WSN research such as routing protocols and storage schemes. To properly evaluate and compare these techniques, it is necessary to study different types of WSN architectures by combining different types of routing protocols and storage schemes.

In diverse wireless sensor networks applications, we can have different types of queries depending on what the applications want to pursue. Researchers have classified the queries in wireless sensor networks in many ways. In this work, we classify the WSN queries based on various criteria. Different network schemes that are combination of different routing protocols and storage schemes are evaluated for different query types in terms of metrics such as the number of transmissions, energy, delay, life span, and local storage capacity. Through analysis and simulation, we determine what kind of routing and storage is most suitable for which particular query characteristics.

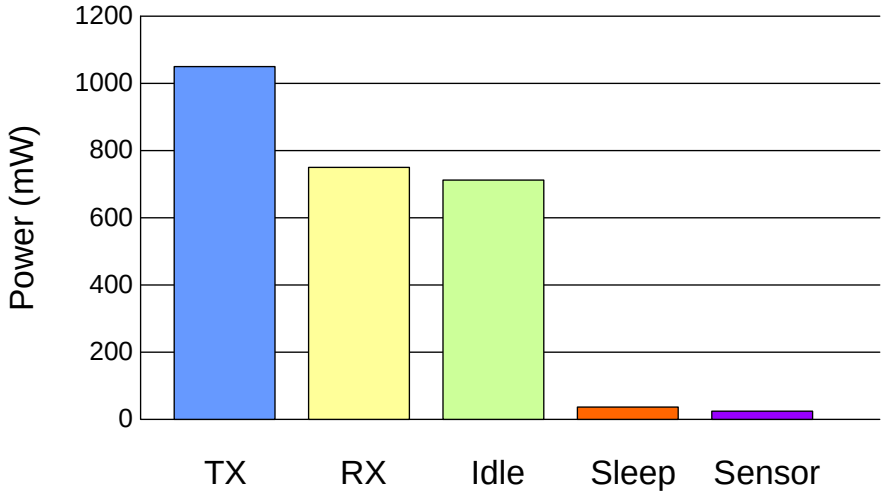


Figure 1.1: Power analysis of Rockwell's WINS sensor nodes [1]

Among the issues of research challenges in wireless sensor networks discussed above, energy constraint is the most critical issue to allow the networks to perform long enough to fulfill their objectives while operating on limited battery resources. Moreover, sensors are mostly deployed in areas where people cannot reach easily and sensors are powered by batteries. Therefore, the batteries cannot be replaced or recharged when they are depleted. As we can see in figure 1.1, most power [1] is consumed when sensor nodes are communicating (transmission(TX) and reception(RX)) and idle. Many researchers have been working on the issue of energy-efficiency of wireless sensor networks by trying to save energy during the most energy consuming tasks (transmission, reception, and idle). This work will mainly focus on energy efficient indexing and data gathering in order to save energy when nodes are communicating with each other.

From the study of characteristics of different network schemes and query types, we consider how to minimize energy usage when queries are disseminated to the network and data are gathered to the user. Queries can be disseminated to all the nodes in the network, or some nodes

[1] We can refer to this graph when comparing energy usage since energy is calculated by multiplication of power and time spent

in the part of the network that users are interested in. Also, the queries can be disseminated by a flooding method or by a routing protocol that considers energy consumption or shortest path to reach. Our interest here is to build a network scheme that minimizes energy consumption when queries are disseminated to all the nodes or some subset of nodes in the network. We consider spatial queries that asks values of some phenomena within a certain area. The main purpose is to save energy by building several local trees that can take care of their own areas and by reducing energy inefficient query dissemination paths. This is possible because most of energy is consumed by wireless communication that sends and receives data [27]. Therefore we try to reduce the number of nodes that consume energy because they are on an unnecessary paths to disseminate queries. This proposed scheme is called sectioned tree. We divide the whole network into several sections, within which local trees are constructed. The local trees are built and organized within the sections so that they can prevent several paths or branches from forwarding queries that consume extra energy.

We also consider energy saving for the other direction of dissemination, which is data gathering. Once queries are disseminated into sensor nodes, they measure and sense what they are supposed to depending on the given queries. There are several ways to report their data to a sink node, where all the sensed data are gathered. The naive way is to report all the data sensed from all the nodes whenever they measure or sense. But in this case, there could be a hot spot node that consumes energy severely, which drives it to deplete the node energy very rapidly. Therefore, in-network processing is proposed that merges similar or related data within a network and report one representative data from a certain group [28]. This technique can save energy since it can reduce the number of transmissions and receptions. Based on this idea, we try to reduce more energy by building an energy-efficient data gathering tree. We prove that finding an optimal solution for data gathering tree is NP-complete, then we propose heuristic algorithms that can build an energy efficient data gathering tree. In the next three sections, we

discuss these of the research problems that we cover in this thesis. Then we give a summary of our contributions in section 1.4.

1.1 Study of queries and network schemes

The ultimate goal for sensor networks is to query information that was collected from sensors. To achieve this goal, we need to try to solve several intrinsic problems, some of which are related to queries. One criterion for classifying query types is based on time. These categories are historical queries, snapshot queries, and long-running queries according to [29].

- Historical queries request a result through data that has been accumulated from the past.
- Snapshot queries request a result from data that is being currently sensed.
- If these queries are extended to keep running for a certain period of time, they are long-running queries.

Besides the time criterion, we need to have more criteria to better characterize each type of query. Many papers [29, 30, 31, 32, 33, 28] discussed query types for sensor network data systems. These papers use different terms to represent similar query concepts, and also they have their own ways to categorize queries. We divide queries into several atomic elements that can be orthogonal to each other as much as possible so that we can analyze the relationship among them.

There could be different architectures needed to maximize the overall performance in sensor networks depending on what kind of query types we use. Among these architectures, one of the new emerging areas is storage architecture for sensor networks. In general, sensor network storage can be classified into local storage, external storage, and data-centric storage depending on where or how to store the sensed data [34]. Local storage stores what each sensor sensed on its own storage space. For external storage, all the data is sent to an external storage and saved on it. Data-centric storage stores data on a different node from the sensing node.

(See figure 1.2.) Each type of storage scheme has certain advantages based on the particular type of query. One goal of this paper is to specify the relationship between type of storage and the query types that each storage supports best.

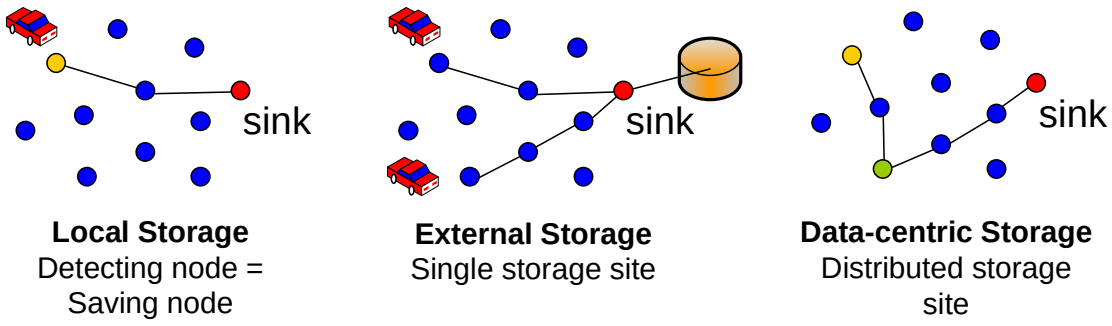


Figure 1.2: Three types of storage

To store data, we need to have not only actual storage, but also a scheme for forwarding the data to the storage from the sensing sources. That means, storage cannot exist by itself without being supported by data dissemination or routing protocols. Models for data delivery required by sensor network applications are classified into continuous, event-driven, and observer-initiated [3]. Continuous data delivery is useful for periodic monitoring systems, which have to send data periodically. Event-driven delivery is that an event triggers an action, which is usually sensing or forwarding data to a sink. Observer-initiated delivery is that sources do sensing, gathering, or forwarding data whenever users send a query.

Besides these classifications, we need to classify architectures for routing protocols so that we can connect data-delivery type, storage type, and query type. Routing protocols can be classified into hierarchical, data-centric, and geographical. Hierarchical routing is a protocol that has more than one level of data transmission and a typical example of this is a clustering structure in which sensor nodes are divided into clusters and each cluster has its own cluster head. Data-centric routing is a protocol that can handle data that is named by attribute-value

pairs. Geographical routing uses physical sensor locations in its protocol and is a neither hierarchical nor data-centric in this paper[2]. We analyze the various combinations of storage, data delivery, and query types to determine which combinations perform best together in sensor networks. We take into account the number of transmissions and energy consumed throughout the whole network, and the delay for packets from source to sink. After we evaluate and compare each combination, we determine what kind of combination is suitable for each particular sensor network application.

1.2 Energy-efficient query indexing

Wireless sensor networks have hundreds or thousands of sensors with physical components and logical functions of being able to sense, store, process, and communicate data. Applications of sensor networks are used for monitoring events and measuring values at places where people cannot reach easily or where a long term sensing task is required. In order to keep sensors alive long enough to fulfill their duty, energy is an important issue, since sensor nodes work depending on limited battery power, which is not feasible or difficult to charge or substitute. Most energy is consumed when sensors are communicating [27]. In-network data processing is useful for reducing the number of transmissions and eventually lowering energy consumption [28]. When we treat spatial queries in sensor networks, we typically use in-network aggregation for data processing to save energy. This kind of scheme uses tree based structures to aggregate data. In this case, when we disseminate a query into the network or gather data from sensor nodes, we would like to find a path that consumes energy least. But, getting an optimal solution for energy efficient topology of wireless sensor networks is an NP-complete problem [35]. Therefore, we suggest a predetermined scheme that supports energy efficient paths heuristically. To accomplish this, we propose an indexing structure that has

[2]Some hierarchical and data-centric protocols also utilize sensor locations.

several sections, each of which has its own local root and a local tree, which does not have a connection with sensor nodes in other sections except through the local root.

Our proposed scheme is based on the following assumptions: 1. A network has sections that divide the network into smaller pieces that can be basic units of area posed by a spatial query. 2. In each section, a local tree with a local root exists and the local root connects to a node in a different section so that they make one big tree. 3. Local trees are built based on either Prim's minimum spanning tree or Dijkstra's shortest path algorithms [36] so that each local tree can consume less energy than when these algorithms are not applied. 4. For query dissemination, we use Minimum Bounding Rectangle (MBR) indexing so that the query can be disseminated into a minimum number of nodes that are in a query area. To evaluate this scheme, we have six different network structures and four different query types. The simulation shows our proposed scheme saves energy when a spatial query is disseminated into the network and data is sent back to the base station.

1.3 Energy-efficient data gathering

Wireless sensor networks consist of sensor nodes equipped with a CPU, memory, wireless communication unit, and multi-functional sensors. In wireless sensor networks, a sink node (or a base station) usually sends a query to source nodes and they send back the sensed data to the sink node based on the query. Wireless sensor networks may have hundreds or thousands of sensor nodes with physical components and logical functions to sense, store, process, and communicate data. Applications of sensor networks are used for monitoring events and measuring values at locations where people cannot reach easily or where a long term sensing task is required. Some application examples include civil structural monitoring [14], habitat monitoring [7], and environmental monitoring such as wildfires [12]. Energy has been an important issue because sensor nodes depend on limited battery power, and it is not feasible or

difficult to charge or replace batteries in many sensor network applications. Therefore, if we can save energy where the most part of it is consumed, which is wireless data communication between sensor nodes [27], then it can eventually prolong lifetime of wireless sensor networks.

In-network data processing is useful for reducing the number of transmissions and hence lowering energy consumption [28]. When we treat spatial queries in sensor networks and especially for queries that request aggregated data such as sum or average, we typically use in-network aggregation for data processing to save energy. An example of this type of spatial query is "report average temperature in a specific region every 10 minutes for 1 month." In this query, a base station (sink node) sends the query once, but it receives aggregated data from the sensor (source) nodes periodically. When data from source nodes are sent back to the sink node, it is common to use a tree based structure for in-network processing and gathering data so that each node processes data from its children to create a partial result . In the example query, which is spatial and periodic, all the sensor nodes transmit data once per period (every 10 minutes), hence each node has to transmit data many times (24 hr $\times$ 6 times/hr $\times$ 30 days $=$ 4,320 times) during the duration (1 month) designated by the query for data gathering. Meanwhile, the query is just disseminated once. Therefore, we would like to focus on energy saving for data gathering, since this constitutes a large percentage of the total energy consumption for these common types of queries.

We get an energy efficient data gathering tree by applying an accurate energy measuring equation on the links between communicating nodes. We build an energy measuring equation that has many parameters in order to reflect real world variations as much as possible. Our energy model is based on the one used in LEACH ([37]) and considers overhearing energy as used in [38]. Most previous work assumed one way transmission without considering the fact that the sender can receive a control packet such as ACK (Acknowledgement). In order to consider energy consumed by control packets, we apply two-way communication as used in IEEE 802.11 or S-MAC ([39]) that has control packets for reliable data transmission and

collision prevention. Based on these assumptions, we prove that our Energy-efficient Data Gathering (EDG) problem is NP-complete by showing that the EDG problem is equivalent to Min-Power Symmetric Connectivity (MSC, [40]). Since the EDG problem is NP-complete, we propose two polynomial time heuristic algorithms that construct more energy efficient data gathering tree structures than existing algorithms such as Energy Conserving Routing Tree (ECRT) [41], Local Optimization (LOCAL-OPT) [41], and the well known Prim's minimum spanning tree (MST) algorithm [36].

1.4 Problems and contributions

1.4.1 Performance evaluation of queries and network schemes

Many kinds of application of wireless sensor networks have different purposes. Because of this reason, the network system should be built based on the fact, such as what phenomena they measure, how frequently they measure and report sensed data, what kind of structure they use, and what kind of algorithms they use. These should be determined in terms of what users emphasize on, such as long term functionality, quick response time, or reliability. Therefore, we classify sensor network queries into individual elements that can represent their own characteristics. After we combine these elements, we can get characteristics of all the possible combination of queries. We also combine different storage types and different routing protocols so that we can determine what kind of network schemes are best for particular query types in terms of number of transmission, energy, end-to-end delay, life span and local storage capacity.

1.4.2 Energy-efficient query indexing

Our contributions affect three features of network structure. First, we have virtual sections, within which local trees are constructed, so as to prevent several branches from forwarding queries or data that might consume extra energy. Second, we use MBR as a spatial query

indexing method through all the connected local trees, so in order to use less energy as queries are disseminated only into the designated area. Third, we use existing tree building algorithm such as Prim's minimum spanning tree in a local tree, so that we have less energy consumption than other tree building methods. Our scheme is feasible and scalable, since the algorithms are calculated at local sub-roots that take care of only limited number of nodes.

1.4.3 Energy-efficient data gathering

Unlike existing research that consider approximation of energy model, we apply an energy measuring equation that considers two-way communication to the energy-efficient data gathering problem. We also consider overhearing energy and idle energy so that we can measure more accurate energy consumption. We define energy-efficient data gathering (EDG) tree problem and prove it as NP-complete by proof by restriction method. Since the problem is proven as NP-complete, we propose heuristic algorithms to find energy-efficient solutions for the EDG tree problem.

1.5 Organization of Thesis

The remainder of the thesis is organized as follow. Chapter 2 describes related work that contain different types of routing protocols and storage schemes. It also describes query dissemination, indexing techniques, and data gathering problems. In chapter 3, we discuss spanning tree problems. We introduce backgrounds of heuristic techniques, genetic algorithm and simulated annealing, and also discuss our techniques for the two heuristic algorithms. Chapter 4 presents the study of sensor network architecture. It includes study of sensor query classification and network schemes, such as routing protocols and storage schemes. Cost analysis and experimental result will be shown for performance evaluation of various query types and network schemes. Chapter 5 proposes sectioned tree that has pre-determined number of square areas and local trees within them to disseminate queries into a particular area energy

efficiently. It also describes how we construct sectioned tree and how we save energy in query dissemination. Chapter 6 defines energy-efficient data gathering problems, proves the problem is NP-complete, and proposes heuristic algorithms to solve the problem. Experiments will be done to compare the proposed algorithms and existing algorithms, such as genetic algorithm, simulated annealing, and several techniques proposed by other work. Finally, we conclude this thesis with future research direction in chapter 7.

CHAPTER 2

RELATED WORK

2.1 Routing Protocols

Routing protocols in wireless sensor networks can be classified into three types, flat routing, hierarchical routing, and location based routing [42]. In the network that uses flat routing, all the nodes play the same role and collaborate each other to perform a sensing task. Some of the examples of flat routing include: Sensor Protocols for Information via Negotiation (SPIN) [43], Directed Diffusion [44], and Rumor Routing [45].

Heinzelman *et al.* proposed the sensor protocols for information via negotiation (SPIN) in [43]. The SPIN protocol disseminates information among sensors and reduces the number of redundant data communication through meta-data negotiations. This protocol uses the idea that neighboring nodes have similar sensed data, and therefore only nodes can disseminate data that other nodes do not have. Intanagonwiwat *et al.* proposed directed diffusion that is also classified as data centric routing protocol. The directed diffusion creates data by attribute-value pairs and eliminates redundancy by in-network aggregation. A base station broadcasts interests, which is a task to be done by the sensor nodes. A gradient is created in each node when it receives an interest, and it decides the flow of interest. Braginsky *et al.* proposed rumor routing, which is a variation of directed diffusion. Directed diffusion can flood queries to the whole nodes in the network, which is not efficient when only a part of nodes are interested in the queries. Rumor routing routes the queries only to a particular region where the nodes detected a specific event. Agents carry the information that a certain node has a particular event and spread it out through the network so that other nodes can know the route to that node.

Some of the examples of hierarchical routing include: Low-Energy Adaptive Clustering Hierarchy (LEACH) [37], Threshold-sensitive Energy Efficient sensor Network protocol (TEEN) [46], Two-Tier Data Dissemination (TTDD) [47], and Virtual Grid Architecture routing [48].

Heinzelman *et al.* proposed another routing protocol, which is known as low-energy adaptive clustering hierarchy (LEACH) in [37]. This is hierarchical clustering protocol that has cluster heads and their members. Since a cluster head communicate with a base station, it needs to aggregate data from its members. The protocol has a rotation algorithm that changes the role of cluster head within a cluster so that they can prevent one node from depleting its energy rapidly. However, a cluster head should report its members' data directly to the base station by one hop transmission. Therefore, a cluster head that is far away from a base station can consume more energy than the one close to the base station, since energy is consumed proportional to the radio range. Manjeshwar *et al.* proposed threshold-sensitive energy efficient sensor network protocol (TEEN) in [46]. Unlike LEACH, TEEN has multi level cluster structure. This enables nodes to consume less energy than LEACH. Data from the lowest level can be reported its cluster head. Then the cluster heads in the lowest level forms second level cluster. In this case, cluster heads in the lowest level are the members of the second level clusters. Now, there is a second level cluster head, which takes care of its members. It is repeated until a base station can form a cluster. By this way, data from the lowest level can be forwarded to the base station by multi-hop routing, which spends less energy than LEACH, since the radio range of cluster heads can be decreased. Luo *et al.* proposed two-tier data dissemination in large scale wireless sensor networks (TTDD) in [47]. Unlike LEACH or TEEN, TTDD uses mobile sink that can travel around stationary sensor nodes. In order to get data from where an event happens, sensor nodes build grid structure that enables two-tier data dissemination. Instead of waiting for a query from a sink, the data source proactively builds grid structure so that they can forward information. Once a mobile sink floods a query into all the nodes in a

cell (lower tier), the nearest dissemination node to the sink forwards the query to a dissemination node that holds the event by multi-hop routing through other dissemination nodes (higher tier). Al-Karaki *et al.* proposed a virtual grid architecture routing in [48]. In this work, they build virtual grid that contains square clusters. Each cluster is called a zone. In a zone, one node is selected as a cluster head among the nodes in the zone. This node is also called a local aggregator, which aggregates data from its zone. Also, these local aggregators form a bigger network, which performs global aggregation. This structure is similar to TTDD in that it uses two level hierarchical structure to report data. Since choosing global points optimally is proven as NP-hard, they proposed an exact algorithm that uses integer linear program formulation and several approximate algorithms.

Location based routing protocols have basically an assumption that nodes know their positions and they can know their neighbors' positions. Therefore, when a node forwards a packet to the other node, they can choose any neighbor geographically that has a certain advantage for considering what each protocol focuses on. Some of the examples of location based routing protocols include: Greedy Perimeter Stateless Routing for wireless networks (GPSR) [49], Geographical and Energy Aware Routing (GEAR) [50], and Geographic Power Efficient Routing in sensor networks (GPER) [51].

Karp *et al.* proposed greedy perimeter stateless routing (GPSR) for wireless networks in [49]. In this work, they proposed greedy forwarding that a node forwards a packet to a node that is geographically nearest node to the destination. This is based on the idea that we can choose a next hop node, which is locally optimal. The assumption for this is that all the nodes know their positions by global positioning system (GPS) or by any other methods that can determine the position. Also, when they build a neighbor list, a node knows the position of its neighbors. Yu *et al.* proposed geographical and energy aware routing (GEAR) in [50]. There are two phases in order to forward packets to all the nodes in a target region. The first phase is to forward the packets towards the target region. When there is a closer neighbor to the destination,

GEAR picks a next-hop node among all the neighbors that are closer to the destination. If it cannot find any node closer to the destination, GEAR picks a next-hop node that minimizes some cost value of this neighbor. The second phase is to disseminate the packet within the region. For the energy-aware neighbor selection, GEAR uses learned cost and estimated cost. This energy-aware neighbor selection is to used for the first phase and recursive geographic forwarding or restricted flooding algorithm is to used for the second phase. Wu *et al.* proposed geographical power efficient routing in sensor networks (GPER) in [51]. The main idea of this protocol is that a source node forwards a packet to a neighbor that is on the route for the sub-destination node if it conserves power. GPER is composed of three basic protocols, which are RouteWithinNeighbors, GPER, and GPER-2. RouteWithinNeighbors enables a sensor to choose the best next node in its radio range. GPER protocol establishes routes for destinations that are not within the radio range of the source node. When a destination node is outside the radio range of a source node, it uses dynamic sub-destination adjustment function so that it can forward a packet to the destination in the end. In the case of no neighbor is closer to the destination than the current node, they adopt planar perimeter algorithm from GPSR. GPER-2 is designed to treat large variations of density in sensor networks.

2.2 Query Classification

Several previous work categorize or classify the queries that can be used in sensor network applications or in monitoring systems. Some of them focus on whole queries and others focus on only aggregate queries.

In TinyDB [28], Madden *et al.* have a taxonomy for aggregate queries. They classify aggregate queries based on four different criteria. Among the aggregate functions such as MAX, MIN, COUNT, SUM, AVERAGE, MEDIAN, COUNT DISTINCT, and HISTOGRAM, these criteria are whether an aggregate is duplicate sensitive or not, whether it is an exemplary

or a summary, whether it is monotonic or not, and how much storage is required for each partial state record.

Bonnet *et al.* discuss three types of queries for monitoring applications, which are historical, snapshot, and long-running in [29]. A historical query is typically aggregate query over historical data. A snapshot query is a query that is interested at a given point of time. A long-running query is over a time interval.

In ACQUIRE, [30], Sadagopan *et al.* classify sensor network queries into four categories. The first criteria is continuous versus oneshot queries. The second is whether a query is aggregate or not. The third is whether a query has sub-queries or not. And, the fourth are queries for replicated data or queries for unique data.

In [31], Kaya *et al.* classify queries based on response characteristics and analyze the sensor network performance in terms of traffic load. The attributes for the classifications are response attempt, response generation method, and response transmission. Based on the response attempt, queries can be divided into continuous and oneshot queries. Response generation method is a criteria to classify queries into simple and complex. By the response transmission, queries can be divided into either concatenated or immediate.

Li *et al.* propose a distributed index especially for multi-dimensional range queries in [32]. They focus on a specific query type, which has two properties, multi-dimensional and range. While they present a framework for cleaning and querying noisy sensors, they classify query types into three forms in [33], which are single source queries (SSQ), non-aggregate queries (SNAQ), and aggregate queries, which can be divided into summary aggregate queries (SAQ) and exemplary aggregate queries (EAQ).

Ratnasamy *et al.* introduce data-centric storage and compared with external storage or local storage in [34, 52]. In [34], they evaluate communication cost regarding the number of transmissions and use asymptotic cost $O(n)$ for message flooding and $O(\sqrt{n})$ for point-to-point routing when *n* nodes are evenly distributed in the square grid topology. They propose

Geographic Hash Table (GHT) that hashes keys into geographic coordinates so that key-value pair data is stored at a specific node, which resides nearest to the hashed coordinates by that hash function and retrieved by a query by the same function. Also, they use GPSR [49] as their underlying routing protocol and use variation of perimeter right-hand rule for finding home nodes, which are nearest nodes of packet destination.

2.3 Query Dissemination

In [28], Madden *et al.* talk about in-network data aggregation, which distributes aggregate queries, and aggregates data while nodes at low tree levels pass their data up to parents. This method can reduce the number of transmissions compared to server-based approach, and eventually saves energy. This in-network data aggregation is one fundamental idea of this thesis. Energy-efficient query dissemination is the second fundamental concept, as we next describe related works to this concept. In [53], Madden *et al.* suggest semantic routing tree, which is an index over a particular attribute that can help a query forwarded to only appropriate nodes that have values within the interval held by its ancestors. This can reduce energy that might be used by query flooding. Soheili *et al.* expand this to a two dimensional semantic tree of spatial query index in [54]. In this work, they use MBR of node's position and the positions of its descendants as an index to disseminate a spatial query into only an index MBR of specific area that intersects the query MBR. This also reduces energy consumption by preventing a query from being disseminated into areas that do not intersect the query MBR. In [55], Coman *et al.* propose an energy efficient query processing method for spatiotemporal region query. They try to disseminate a query only to a small subset of nodes in the network and the spatial query processing is performed closer to the data sources.

In [56], Beaver *et al.* talk about location aware routing for data aggregation in sensor networks. Unlike the traditional tree building method, they propose an algorithm that nodes

can choose their parents to be aware of the same group so that it can reduce the number of messages against a query that has a group-by clause.

When we consider energy consumption in data communication of sensor networks, transmitting a packet is important since it consumes most energy at each node. Several papers work on energy minimization in broadcasting packets or wireless data communication. In [57], Yang *et al.* propose a dynamic query-tree that adjusts tree structure dynamically. They consider energy balancing when sensor nodes broadcast data and assumes only leaf nodes can measure data and non-leaf nodes can only forward messages. In [58], Čagalj *et al.* talk about energy-efficient broadcasting in all wireless networks. They show that the minimum broadcast cover (MBC) problem is NP-complete when an arbitrary set of nodes is given.

It is also important to use a proper energy model to evaluate energy consumption in sensor networks. In [59], Khan *et al.* consider only transmission energy when they treat energy-efficient routing schemes in wireless sensor networks. In [41], Buragohain *et al.* consider both transmission and reception energy but they ignore the energy change depending on the distance between two nodes. They propose power aware routing for sensor databases by adopting minimum degree spanning tree (MDST) and showing it is an NP-complete problem. Basu *et al.* emphasize the importance of overhearing in wireless sensor networks in [38]. Their energy model includes energy for transmission, reception, and overhearing and consider RF propagation path loss, which we follow as our energy model.

2.4 Data Gathering

Data communication consumes most of the energy that each node dissipates in sensor networks. There are two directions for communication: sending a query and gathering the data for query results. In general, a query is disseminated from a base station to the sensor field, and data are gathered from the sensor field and conveyed to the base station. Here, we can

consider two problems, broadcasting and data gathering. In [60], Wieselthier *et al.* discuss energy-efficient broadcast and multicast trees. Before the problem is proven as NP-complete, they proposed a heuristic algorithm, Broadcast Incremental Power (BIP). On the other hand, the data gathering tree problem is regarded as simpler than the broadcasting problem. In [61], Hong *et al.* proposed two approximate algorithms, Cumulative Increment Algorithm (CIA) and Cumulative Sum Increment Algorithm (CSIA) to solve wireless broadcasting problem that uses Cooperative Wireless Advantage (CWA), which is also shown to be NP-complete.

For data gathering, in [38], while Basu *et al.* emphasize the effect of overhearing energy, they consider both the data gathering problem and the data dissemination problem. In their data gathering problem, they assume simple media access control (MAC) protocol, where there is no IEEE 802.11 carrier-sense multiple access protocol with collision avoidance (CSMA/CA). But in common practice, the two way communication MAC protocol such as IEEE 802.11 or S-MAC [39] is used frequently for reliable data transmission and collision prevention. In this thesis, we consider this so that the data gathering tree problem can better reflect reality. In [62], Liang *et al.* focus on a data gathering problem whose size of data could vary depending on a query. This applies to queries that typically do not involve aggregation. Cheng *et al.* define strong minimum energy topology in wireless sensor networks and study a problem that assigning transmit power to each sensor so that a topology containing only bidirectional links is strongly connected in [35]. Then they prove the problem is NP-complete and propose heuristic algorithms of performance ratio of 2 to approximate the problem.

Power or radio range assignment to minimize the total power consumption by the networks has been researched in the area of ad hoc networks. Kirousis *et al.*, in [63] study MIN RANGE ASSIGNMENT problem and prove that the problem is NP-hard in 3-dimensional space. In [64], Clementi *et al.* prove that the MINIMUM RANGE ASSIGNMENT problem is NP-hard in 2-dimensional space. In [40], Althaus *et al.* propose MIN-POWER SYMMETRIC

CONNECTIVITY problem that is similar to the previous two problems and propose approximation algorithms.

CHAPTER 3

SPANNING TREE PROBLEMS

3.1 Introduction

A problem of minimum spanning tree is to find a minimum sum of costs on edges of the spanning tree. Well known methods to solve a minimum spanning tree problems include Prim's algorithm [65] and Kruskal's algorithm [66]. These algorithms can solve a minimum spanning tree problem in $O(E \log E)$ time[36]. Besides a minimum spanning tree problem, we have many variants of it. Many of these problems fall in the criteria of NP-hard problem depending on their constraints. Degree constrained spanning tree [67] remains NP-complete if we try to find a tree in which no node has degree more than 2 incidents. Maximum leaf spanning tree problem [68] has its constraints, which is the number of leaves. Bounded diameter spanning tree problem [68] is to find a spanning tree in which the sum of the weights of the edges is bounded by an integer B and there is no simple path with more than D edges. Capacitated spanning tree problem [69] has a constraint of the capacity of edges.

Since many of the spanning tree problems are in NP-hard, researchers have adapted some of the heuristic techniques, such as genetic algorithms and simulated annealing. In this chapter, we study several techniques that solve variants of spanning tree problems by applying several heuristic techniques. In the following two sections, we explain briefly two methods that are used in this thesis, simulated annealing and genetic algorithm. In the third section, we introduce several techniques that were used to solve many kinds of minimum spanning tree problems by these techniques.

3.2 Genetic Algorithm

3.2.1 Biological Background

Genetic algorithm was first developed by Holland in 1975. In his book ”Adaptation in natural and artificial systems”, he described how to apply Darwin's theory of natural evolution to optimization problems. The genetic algorithm, therefore, is directly derived from the principles of genetics and natural evolution [70]. This algorithm is based on a stochastic method that depends on probability distribution. The algorithm is not guaranteed to find the optimal solution, instead it can find a solution which is quite a bit close enough to the optimal solution. Therefore, this genetic algorithm is useful and used to solve NP-complete problems by suggesting ”acceptably good” solution.

In real world, a cell is the structural and functional unit of all living organisms. In the cell, there is a cell nucleus that contains chromosome, in which all the genetic information are stored. Properties of species that determine the characteristics of an individual are encoded in genes. Each gene encodes a trait, such as eye color. Alleles are possibilities of the genes for the particular property. The set of all the genes of a specific species is called genome. Each gene has its own position and it is called locus. A complete genetic sequence on one set of chromosomes is called genome. The entire combination of genes in genome is called genotype. During reproduction, parents create their new offspring by recombination (or crossover). Genes combined from the parents create a chromosome that has a combination of genetic information from the two. Also, the offspring can be mutated, which means a small part of information in chromosome is changed because of copying errors caused by many reasons.

3.2.2 Basic of Genetic Algorithm

The genetic algorithm mimics this biological sequences. The algorithm starts with random generation of population of n chromosomes [70, 71, 72]. It evaluates each chromosome in the initial population by using the fitness function. Now, new population is created by re-

peating four steps, selection, crossover, mutation, and accepting. In the first step, selection, the algorithm chooses two parent chromosomes based on the fitness function. In this case, the better value from the fitness function, the more chances to have to be chosen as parents. The second step, crossover, selects genes from parents chromosome with a crossover probability, then cross over the parents chosen from the selection step to create a new offspring chromosome. At the third step, mutation takes place by giving mutation probability to each locus of new offspring chromosome. Now, the newly created offspring join the existing population. After this, new generated population will be considered as the algorithm repeats the previous four steps. At the end of this repetition, there is a terminating condition, so that the process can stop when the condition is satisfied.

Evolutionary algorithm includes genetic algorithm, evolution strategies, and evolution programming [73]. Since the discussion about the difference between evolutionary and genetic algorithm is beyond the scope of this thesis, we just use the main concept of two algorithms interchangeably. Figure 3.1 shows overview of sequences of genetic algorithm.

Procedure 1: Genetic Algorithm

```
1: begin
2:     initialize population with randomly generated candidate solutions
3:     evaluate each candidate solution
4:     while (termination condition not satisfied) do
5:         select parents
6:         crossover pairs of parents to create a offspring
7:         mutate the offspring
8:         evaluate the new candidate
9:         replace  the new candidate generating a new population
10:    end while
11: end
```

Figure 3.1: Pseudocode of genetic algorithm

Raidl *et al.* proposed edge sets to represent spanning trees as suitable codes for evolutionary algorithm in [74]. This spanning tree encoding technique is applied to solve NP-hard problem degree constrained minimum spanning tree so that their technique performs well in initialization, recombination, and mutation in terms of locality, heritability, and computational efficiency. Chou *et al.* studied the factors of genetic algorithm that influence performance for solving spanning tree problem in [75]. Since there are many variations of techniques in each step, such as encoding methods, crossover methods, and mutation methods, they experimented combinations of the techniques and evaluated the performance to solve degree constrained minimum spanning tree problem. By their result, the best solution quality can be performed when they combine determinant encoding, uniform crossover, and exchange mutation.

3.2.3 Genetic Algorithm in This Thesis

In chapter 6, we use the genetic algorithm to solve our problem, energy-efficient data gathering tree (EDG-tree) problem. This algorithm will be compared to many other algorithms. In this section, we describe techniques used in our genetic algorithm in detail.

- Initialization Phase

  We create 20 trees as chromosomes for initial population. One is from Prim's minimum spanning tree [65] and the others are from random generated trees. The reason why we choose Prim's minimum spanning tree algorithm is that it is proven as 2-approximation algorithm for the Min-power Symmetric Connectivity (MSC, [40]) problem, which will be discussed in chapter 6. Traditionally, the population size is also generated randomly, however, 20 to 30 initial candidates are considered as good population size [74].

- Evaluation Phase

  In order to evaluate each chromosome, we use our energy model that summate cost on edges as a fitness function.

- Selection Phase

In order to select suitable parents chromosomes, many methods have been introduced, such as roulette wheel, Boltzman selection, tournament selection, rank selection, best selection, and steady state selection as described in [76]. Among the methods, we choose two chromosomes as parents that are randomly chosen from top 20% pool of the initial population.

- Crossover Phase

  In order to implement crossover, we have also many methods such as one point, two points, uniform, arithmetic, and heuristic method. Since we are using tree representation, we adopt edge-set representation and edge crossover that was proposed by Raidl in [77]. In the first step, we find all the common edges from both parents trees. These edges will definitely form a new offspring tree. Next, we find edges that belong to either of parents, not both of them and make it a temporary edge set. Using the disjoint set algorithm as described in [36], we try to build an offspring tree by choosing one edge by edge from the temporary edge set so that the tree always satisfies tree building condition.

- Mutation Phase

  Mutation has also many techniques, such as flip bit, boundary, non-uniform, uniform, and Gaussian method. In our algorithm, we simply choose one edge that belongs to the offspring tree, then replace it with one that belongs to the temporary edge set. If an edge does not create a tree, then the random selection will happen until an offspring tree is created. But if there is no edge that can replace an existing one, mutation will not occur.

3.3 Simulated Annealing

3.3.1 Background

Thermodynamic annealing [78, 79] is a process for glass or metal to have a solid structure of their crystals. The glass or metal is heated up to a certain temperature at which they can be

melted into the liquid state. In this state, the molecules of hot glass or metal are free to move about. After that, if the temperature drops rapidly, these molecules get out of equilibrium and eventually make the crystal have many defects. On the other hand, if the temperature drops slowly, they form a highly ordered and pure crystal. Also, the molecules of a crystal solidify into a minimal energy state. Simulated annealing is an applied algorithm based on simulating the real thermodynamic annealing.

Simulated annealing [80] can solve combinatorial optimization problems. It was initially invented by Kirkpatrick *et al.* in [78] and applied to a thermodynamic annealing problem. This is one of the techniques that can get heuristic solution when the exact optimal solution is not feasible because the problem is intractable to solve. The simulated annealing method is based on stochastic techniques. It also has an algorithm that can try to escape from local minima and eventually get a global solution. It chooses not only downhill moves [1] obviously but also uphill moves depending on a stochastic value, which is compared to a factor, *Temperature*. This means even though the algorithm meets a new value, which is not as good as the current value, it permits the value as the new current value with a probability related to the Temperature. This enables the algorithm to escape from a local minima. The Temperature mimics temperature used in the real world when annealing is applied to glass or metal.

3.3.2 Simulated Annealing in This Thesis

We modified the pseudocode of simulated annealing for the traveling salesman problem (TSP)[79] to apply it to solve energy-efficient data gathering tree (EDG-tree) problem that will be discussed in Chapter 6.

The simulated annealing algorithm works as follows: Once an initial tree is determined randomly, based on query types, the total energy consumed by the whole network is calculated. This value is saved as current cost that is later compared to a new cost from a new tree.

[1] If we consider minimization problem.

We set initial Temperature value as an average energy that each node consumes. Now, for a pre-determined number of iterations, the actual annealing step is executed with a decreasing Temperature by each iteration. The decreasing rate of the Temperature can be adjusted by a factor, which mimics how fast temperature drops in the real world.

Procedure 2: innerLoop

```
begin
 1: for i ← 1 to n_temps do
 2:    temperature ← factor * temperature
 3:    for j ← 1 to local_iteration do
 4:       current_cost ← cost evaluation for T_C
 5:       //try to replace an edge with a random edge
 6:       T_T ← pickOneEdge()
 7:       trial_cost ← cost evaluation for T_T
 8:       delta ← current_cost - trial_cost
 9:       if delta > 0 then // better cost
10:          T_C ← T_T
11:          n_swaps ← n_swaps + 1
12:       else
13:          p ← random real number between 0 and 1
14:          m ← e^( delta / temperature )
15:          if p < m then
16:             T_C ←T_T
17:             n_swaps ← n_swaps + 1
18:          end if
19:       end if
20:       exit when n_swaps > global_iteration
21:    end for
22: end for
end
```

Figure 3.2: Pseudocode of simulated annealing

In each annealing step, with the same Temperature, a new tree is constructed by replacing one existing edge with a new edge. The energy for the new tree is calculated to get a new cost. The construction for the new tree is described in more detail in the next paragraph. If the new

cost is cheaper than the current cost, the new edge replaces the existing one and eventually sets a new tree (line 9∼11 in Figure 3.2). If the new cost is worse than the current cost, still we have a chance to allow the new tree to be a current tree. Line 13 generates a random number between 0 and 1 and set it to p, then line 14 generates a value that follows decreasing exponential function so that the value can be between 0 and 1 and set it to m. If we have a large $delta$ value, m will be close to 0, and a new tree with higher cost is unlikely to be accepted since the probability of p being less than m decreases. Therefore, line 15 condition may accept the new tree even though the new cost is worse than the current one, and if the condition is not satisfied, the new tree will be discarded.

When the procedure `pickOneEdge` is called by line 6 in the procedure `innerLoop` (Figure 3.2), the procedure `pickOneEdge` (Figure 3.3) will replace one existing edge with another edge, which is not in the current tree. To replace one edge (u, v), first, we have to select a new edge (i, j) from a set of edges that are not currently participating the tree, $(i, j) \in E_T^c$, where $E' = E_T \cup E_T^c$. Second, the new edge should connect the two trees that were caused by absence of the old edge, (u, v). It is straightforward that these two conditions guarantee not creating a cycle. The pseudocode of `pickOneEdge` is represented in Figure 3.3: Initially, we have $E_T \subset E'$ and $|E_T| = |V| - 1$ (line 1). An edge (u, v) is then chosen randomly (line 2) and E_T is newly set as $E_T - \{(u, v)\}$, which makes $|E_T| = |V| - 2$ (line 3). V is now separated into two subsets V_1 and V_2 ($V_1 \cup V_2 = V \; and \; V_1 \cap V_2 = \phi$), each of which forms two different trees[2]. Then, we choose (i, j) randomly, which satisfies, $(i, j) \in E_T^c$, where $(i \in V_1 \; and \; j \in V_2) \; or \; (i \in V_2 \; and \; j \in V_1)$, and $(u, v) \neq (i, j)$ (line 5). This can be implemented by using disjoint set as described in [36]. The new edge (i, j) is added to E_T (line 6) and finally, the new tree T is returned to the procedure `innerLoop` (line7).

[2]We allow one vertex tree

Procedure 3: pickOneEdge

begin
1: E_T is ready // $|E_T|=|V|-1$
2: pick (u,v) randomly, which satisfies (u,v) $\in E_T$
3: take out (u,v) from $E_T \leftarrow E_T - \{(u,v)\}$ // $|E_T|=|V|-2$
4: // V is now separated into V_1 and V_2 due to taken out (u,v)
5: choose (i,j) from E_T^c , which satisfies
$\{(i \in V_1$ and $j \in V_2)$ or $(i \in V_2$ and $j \in V_1)\}$ and $(i,j) \neq (u,v)$
6: $E_T \leftarrow E_T \cup \{(i,j)\}$
7: **return** $T(V,E_T)$
end

Figure 3.3: Pseudocode of replcaing one node with another

3.4 Summary

In order to solve spanning tree problem that is proven as NP-complete, we use global optimization techniques, genetic algorithm and simulated annealing to compare them with our own algorithm. Among many specific techniques in each step of them, we implement genetic algorithm and simulated annealing so that they can represent their own properties. These algorithms will be compared with other techniques in chapter 6.

CHAPTER 4

SENSOR NETWORK ARCHITECTURE

4.1 Introduction

In this chapter, we study characteristics of many kinds of network schemes and query types that are used in different applications of wireless sensor networks. First of all, we classify sensor network queries into several criteria, time, aggregation, filter, dimension, and replication. These criteria are divided into sub-criteria so that each of them represents fundamental characteristics. We also discuss different types of storage schemes and routing protocols and study their characteristics. Now, we have many types of queries that are combined from sub-criteria and different types of network schemes that are combined from different storage types and routing protocols. We analyze worst case cost in terms of the number of transmissions, energy consumption in a hot spot node, end-to-end delay, life span and local storage capacity. In order to know the cost for the average case, we simulate the query dissemination, aggregating data, storing sensed data, and report them. We measure three major metrics, the number of transmissions, energy consumption, and end-to-end delay through the simulation.

The remainder of this chapter is organized as follows. Section 4.2 classifies queries into several criteria and discuss the characteristics of each criterion. Section 4.3 studies existing storage types and routing protocols. In section 4.4, we analyze cost for the worst case in terms of different metrics. Section 4.5 shows the result of simulation that represents average case. Finally, section 4.6 summarizes this chapter.

4.2 Sensor Query Classification

Sensor network queries can be classified into five different groups. The criteria we use for classification are time, aggregation, filter, dimension, and replication. We discuss each of these criteria next.

Table 4.1: Query classification and different terms in several papers

criteria	classification	[29]	[30]	[31]	[32]	[33]	[28]
time	one-shot	snap-shot	one-shot	one-shot			
	continuous	long-running	continuous	continuous			
	limited-range	historical					
aggregation	spatial-aggregate		aggregate	complex		aggregate	aggregate
	temporal-aggregate		aggregate	complex		aggregate	aggregate
	aggregate-summary		aggregate	complex		summary aggregate	summary
	aggregate-exemplary		aggregate	complex		exemplary aggregate	exemplary
	non-aggregate		non-aggregate			non-aggregate	
filter	filtering			complex	range		
	non-filtering			simple			
dimension	one-dimensional		simple	simple		single source	
	multi-dimensional		complex		multi-dimensional		
replication	replicated		queries for replicated data				
	unique		queries for unique data				

1) Time range when a query is applied:

- If a user just wants to know data at a specific moment, the query is posed just once and data is returned only for the particular moment. This is called *one-shot* query since the start time and end time for the time range is the same.
- If a user wants to know data periodically from a particular moment (usually from now on) to logically infinite time, it is called *continuous* query. In this case the start time can be 'now' and the end time is 'infinite' if the user wants to know the result starting at the time the query is posed.

- When a *continuous* query has a particular end time other than infinite, it can be a query that has *limited range*.

2) *Aggregate* queries versus *non-aggregate* queries. *Aggregate* queries can further be classified into spatial-temporal and exemplary-summary.

- If a user wants to know an average or any other aggregate data from several sensors that are in a certain region, this is called a *spatial-aggregate* query. If a user wants to know an average data for a certain period from one sensor, it is called *temporal-aggregate* query.
- AVG, SUM, and COUNT can be classified as *summary-aggregate* since they need to compute some properties over the entire set of values. MIN and MAX are called *exemplary-aggregate* since they return a representative value from the set of all values [28].

3) Filter versus non-filter. Based on this, we can have *filtering* query, which returns sensor data only if it is within a specified range and *non-filtering* query, which returns all sensor data. When a range is given in a query, it filters out data that are out of range and gives results only satisfying the range condition back to the user.

4) Number of types of sensed data. *One-dimensional* query requests only one type of sensed data, and *multi-dimensional* query requests two or more types of sensed data. For example, a query asking only for temperature is one-dimensional and a query asking for both temperature and brightness is multi-dimensional.

5) *Replicated* or *unique*. If multiple sensors sense the same or similar data, the query can return replicated data. If each sensor measures unique data, the query gets unique data as a result.

For all the classifications discussed so far, terms used in other referenced papers, and their matching relationship is summarized in Table 4.1. The first two columns give our classi-

fication criteria, and the other columns are for listing different terms for the same concepts as used in several papers.

Some of the criteria are orthogonal since they have no dependency on each other. For example, criteria for filtering and dimension are orthogonal, so they can be used in a query at the same time. For example, ”give me a sensor number with values of temperature and humidity whose temperature is between 50 and 60 and whose humidity is between 20 and 30” is a mixture of filtering query and multi-dimensional query.

A taxonomy for aggregate queries is discussed in [28]. The first is whether a result is sensitive or not regarding to duplicates. MAX and MIN functions are duplicate insensitive and COUNT, SUM, and AVERAGE are duplicate sensitive. The second is exemplary versus summary aggregation. The third is whether the aggregate is monotonic or not. When a new value is added on the previous data set, if the result is monotonically increasing or decreasing, it is monotonic aggregate, otherwise, it is not monotonic. COUNT,MAX,and SUM are monotonically increasing and, MIN is monotonically decreasing. But AVG is not monotonic since the result can be fluctuating whenever a new value is added. The fourth depends on partial state records, which are related to the amount of state. For example, a partial AVG record consists of a pair of values, COUNT and SUM, while a partial COUNT record needs only a single value. There are also distributive, algebraic, holistic, unique, and content-sensitive aggregates.

4.3 Storage Types and Data Routing Schemes

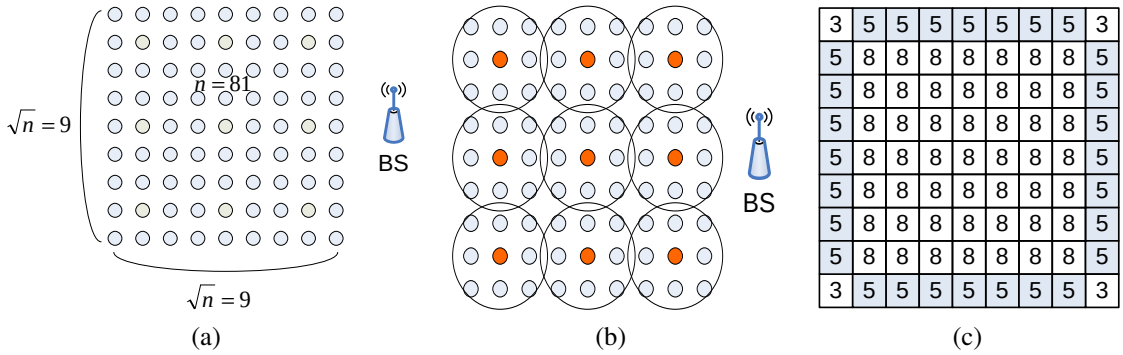
There are several schemes for data delivery in the sensor network. We can classify the schemes based on when and how the data should be delivered to the sink. The top class would be classified into continuous, event-driven, and observer-initiated in terms of the data delivery required by the application [3].

- Continuous: data is generated and forwarded to a sink continuously according to the task that is already embedded in the sensor network or disseminated into the network initially when the network is initialized. This is useful for periodic monitoring application.
- Event-driven: data is generated or triggered by an event and is forwarded to a sink. This can be used in any situation that needs event-driven results such as managing parking spaces.
- Observer-initiated: data is generated by the way that is specified in a query, which is disseminated into the sensor network when a user wants it.

We now show the relationships between data dissemination types and storage types. Storage types are generally classified into three categories: local storage, external storage, and data-centric storage [34]. Local storage means that sensor nodes store what they sensed on their own storage space. Usually the storage is a small amount of cache or memory equipped on the sensor node. External storage is that all the data sensed from sensors is sent to an external storage and saved generally on a fixed hard drive attached to a computer. Data-centric storage requires that data are assigned names based on the type of data produced by the sensors, and stored at a particular node by a predetermined way, for example, geographic hashing function [34]. Determining where to store is based on the 'name' rather than specified by a node address.

So far, many data routing schemes have been proposed. We classify them into several categories that have intrinsic properties. Those are geographical, hierarchical, and data-centric data routing schemes.

- In geographical routing, nodes know the spatial position of their neighbors.
- Hierarchical routing provides a structured level on the network topology such as clustering or grid-based virtual topology [81].
- Data-centric routing uses named-data to route a packet unlike the traditional routing scheme that uses IP address. Typically, the type of query and data includes attribute-value pairs.



(a) (b) (c)

Figure 4.1: Sample topologies for analysis (a) Basic geographic topology (b) Hierarchical topology (c) Number of neighbors

We now discuss the routing schemes that our analysis in section 5 is based on. In geographical data routing, we assume a simple topology that has n homogeneous sensor nodes evenly distributed in a square area as shown in Figure 4.1(a). All the nodes have abilities to sense phenomena, relay packets, and aggregate data from neighbors. Among them, there are source nodes that send sensed data and sink nodes that receive the data and connect to a base station, which is outside the sensor network. In hierarchical routing, we assume a simple model that looks like LEACH (Low-Energy Adaptive Clustering Hierarchy [37]), but has an ability for cluster heads to communicate with neighboring ones in a multi-hop way, and can finally forward packets to the base station. Basic sensor nodes deployment is the same as in a geographic data dissemination scheme (see Figure 4.1(b)). In data-centric routing, we use a scheme that is similar to Directed Diffusion [82], but the assumption is that the scheme supports not only the typical scenario of vehicle detection, whose interest and data has attribute-value pairs, but also the scenario for general sensing tasks, such as measuring temperature or humidity. It also uses the same topology as the one used in geographical scheme.

If we combine data delivery types and storage types, we can find several good matching pairs as shown in Table 4.2. The first column represents data delivery types and the second

column is for the three storage types. The third column, relevance, means how well the first column and the second column correlate when their basic properties are considered.

Table 4.2: Matching for data delivery types and storage types

data delivery	storage	relevance	what to send	typical examples	type
continuous	local	low	x	x	-
	external	high	raw data, answer	send me the temperature	*A*
	data-centric	low	x	x	-
event-driven	local	low	x	x	-
	external	high	raw data, answer	send me a data if *a* is occupied	*B*
	data-centric	low	x	x	-
observer-initiated	local	high	raw data, answer	varies	*C*
	external	high	raw data, answer	varies	*D*
	data-centric	high	raw data, answer	varies	*E*

We can see that both continuous and event-driven data delivery are well matching with external storage type. The other two storage types are not appropriate for the two data delivery types in that the main purpose of continuous and event-driven data delivery is to send data to a base station so that the data can be handled in a real time way. But, for the observer-initiated type, it is well matching with all three storage types. Therefore, we consider only observer-initiated case for cost analysis in section 4.4.

We now give some definitions of terms that are used in the remainder of this paper.

- A *task* determines a job that the sensor network has to do. For example, a task could be to measure temperature or humidity or both. It can be installed on the network before the sensors are deployed or it can be forwarded by a user when necessary. It is usually a long-running query, which is periodically continuous.
- *Raw data* is collected from sensors without being modified, such as pure measurement from a sensor.
- *Query* is a question or an interest to get a result from the sensor database. It is distinguished from task since it is more specific than task.

- *Answer* is a response from a specific query. It is also called result in the general database area.

4.4 Cost Analysis

In this section, we evaluate four sample queries in terms of three metrics based on the combinations of storage type and routing schemes. The followings are the three metrics that we consider:

- Number of transmissions: we count the number of transmissions in the network when a query is issued from a sink until the sink gets an answer from the sources.
- Energy: we model the amount of energy dissipated in the network after a query is issued from a sink until the sink gets an answer from the sources. Total amount of energy dissipated in the network and energy dissipated in a particular node are considered.
- Delay: we model the time taken after a query is issued until the sink gets an answer.

From section 4.2, we have 36 combinations of queries by considering the criteria time, aggregation, filter, and dimension. The four cases of combining, one-shot and temporal aggregation are excluded because of contradiction. Among those 32 combinations, we pick four cases to analyze with storage types.

- Case 1: one-shot, non-aggregate, non-filtering, and one-dimensional. An example is ”give me the value of current temperature at sensor #1”.
- Case 2: limited-range, spatial-aggregate, non-filtering, and one-dimensional. An example is ”give me the value of average temperature at sensor #1 through #9 every 1 minute for the next one hour”

- Case 3: limited-range, temporal-aggregate, filtering, and one-dimensional. An example is "give me the value of average temperature of all the values sensed at sensor #1 every 1 minute for next one hour if it is greater than 70."
- Case 4: continuous, spatial-aggregate, filtering, and multi-dimensional. An example is "give me the values of average temperature if it is greater than 70 and average humidity if it is between 20 and 30 at sensors #1 through #9 every 1 minute from now on."

The sample network that is used in cost analysis satisfies the following conditions:

- All the sensors are static and homogeneous
- Transmission range includes all the one-hop nodes as neighbors
- Sources can be any sensor nodes in the network and a sink can be any node either at one of the four corners or at one of the four sides.
- The link between two nodes is robust, so there is no retransmission
- Sensor nodes have enough storage capacity so as to hold what they sense without loss

For the cost analysis of the three metrics, we use n sensor nodes evenly distributed in the grid networks for geographical routing scheme, which forms $\sqrt{n}$ by $\sqrt{n}$ square grid topology as shown in Figure 4.1(a). A transmission range of a node can cover only one-hop neighbor. Therefore, four nodes at the corners have 3 neighbors, $4(\sqrt{n}-2)$ nodes at the sides have 5 neighbors, and the other inner nodes have 8 neighbors. Figure 4.1(c) shows the number of neighbors at nodes in the geographical routing network. A base station that connects sensor network and outer network is reachable by one hop from the sink. For the hierarchical routing scheme, the topology model is basically the same as in geographical one, and every cluster head has eight cluster members as shown in Figure 4.1(b). A transmission range for a cluster head is to cover its neighbor cluster heads (up to eight) so that they can communicate with each other.

Table 4.3 shows the dominant factors when the corresponding query classification is included in a particular query. t_D is a duration of limited range that specifies when a query

is effective. t_P is time for one period and t_L is the average life time of sensors. n_S is the number of sensors that participate in an aggregate query and n_d is the number of dimensions. p_i represents a probability that the dimension of i is satisfied. Now, we analyze each metric one by one and we only consider the worst case scenario so that each result has upper bound.

4.4.1 The number of transmissions

Table 4.3: Query classification and its dominant factor

classification	characteristics	dominant factors
one-shot	$x = t_1$	.
limited range	$t_1 \leq x \leq t_2$	$\frac{t_D}{t_P}$
continuous	$x \geq t_1$	$\frac{t_L}{t_P}$
non-aggregate	.	.
spatial-aggregate	.	n_S
temporal-aggregate	.	.
filtering	.	p_i
multi-dimensional	.	n_d

We consider two routing schemes, geographical and hierarchical for the metric number of transmissions. All three storage types are evaluated in geographical routing, and local storage and external storage are considered in hierarchical routing since data-centric storage is not suitable for hierarchical clustering scheme. In Table 4.4, 'Q' stands for sending query (or task), 'S' for storing sensed data, 'A' for aggregating data, and 'R' for sending answers. 'Geo' stands for geographical routing scheme and 'Hier' for hierarchical scheme. 'LS', 'ES', and 'DCS' stands for local storage, external storage, and data-centric storage, respectively.

The case 1 query is about a simple query that is composed of one-shot, non-aggregate, non-filtering, and one-dimensional. We can calculate the number of transmissions by considering the steps for sending query (or task), storing sensed data, aggregating data, and sending answers with respect to the three types of storage, local, external and data-centric. Since the geographical routing scheme knows where to send packets, the maximum number of transmis-

sions for the peer-to-peer communication is O($\sqrt{n}$). In Figure 4.4, for the case of local storage in geographical routing, the maximum number of transmissions is $2\sqrt{n}$, which is the sum of the number of transmissions that are needed to send a query and sending an answer. It also takes $2\sqrt{n}$ for external storage, but the $\sqrt{n}$ in row Q is for sending task instead of query. And $4\sqrt{n}$ is for the data-centric storage case. The first $\sqrt{n}$ in the row Q of the data-centric storage column is to send a task to the designated node and the last $\sqrt{n}$ is for sending actual query by peer-to-peer unicast way. For the local storage and external storage in hierarchical routing, it is $2(\sqrt{m}+1)$, where m is the number of cluster heads in the network.

Table 4.4: Number of transmissions in case 1 and 2

case	Case 1					Case 2				
routing	Geo			Hier		Geo			Hier	
storage	LS	ES	DCS	LS	ES	LS	ES	DCS	LS	ES
Q	$\sqrt{n}$	$\sqrt{n}$	$\sqrt{n}$, $\sqrt{n}$	$\sqrt{m}+1$	$\sqrt{m}+1$	$n_S\sqrt{n}$	$n_S\sqrt{n}$	$n_S\sqrt{n}$, $\sqrt{n}$	$\sqrt{m}$+1	$\sqrt{m}$+1
S	x	$\sqrt{n}$	$\sqrt{n}$	x	$\sqrt{m}+1$	x	$\frac{t_D}{t_P}n_S\sqrt{n}$	$\frac{t_D}{t_P}n_S\sqrt{n}$	x	$n_S\frac{t_D}{t_P}(\sqrt{m}$+1)
A	x	x	x	x	x	$\frac{t_D}{t_P}(n_S$-1)$\sqrt{n}$	x	x	$(n_S$-1)$\frac{t_D}{t_P}$	x
R	$\sqrt{n}$	x	$\sqrt{n}$	$\sqrt{m}+1$	x	$\frac{t_D}{t_P}\sqrt{n}$	x	$\frac{t_D}{t_P}\sqrt{n}$	$\frac{t_D}{t_P}\sqrt{m}$	x

Table 4.5: Number of transmissions in case 3

routing	Geo			Hier	
storage	LS	ES	DCS	LS	ES
Q	$\sqrt{n}$	$\sqrt{n}$	$\sqrt{n}$,$\sqrt{n}$	$\sqrt{m}+1$	$\sqrt{m}+1$
S	x	$\frac{t_D}{t_P}\sqrt{n}$	$\frac{t_D}{t_P}\sqrt{n}$	x	$\frac{t_D}{t_P}(\sqrt{m}$+1)
A	x	x	x	x	x
R	$p_i\sqrt{n}$	x	$p_i\sqrt{n}$	$p_i(\sqrt{m}$+1)	x

Table 4.6: Number of transmissions in case 4

routing	Geo			Hier	
storage	LS	ES	DCS	LS	ES
Q	$n_S\sqrt{n}$	$n_S\sqrt{n}$	$n_S\sqrt{n}$,$n_S\sqrt{n}$	$n_S(\sqrt{m}$+1)	$n_S(\sqrt{m}$+1)
S	x	$n_d\frac{t_L}{t_P}n_S\sqrt{n}$	$n_d\frac{t_L}{t_P}n_S\sqrt{n}$	x	$n_d\frac{t_L}{t_P}n_S(\sqrt{m}$+1)
A	$n_d\frac{t_L}{t_P}(n_S$-1)$\sqrt{n}$	x	x	$n_d\frac{t_L}{t_P}$	x
R	$\frac{t_L}{t_P}(p_1$+$p_2)\sqrt{n}$	x	$\frac{t_L}{t_P}(p_1$+$p_2)\sqrt{n}$	$\frac{t_L}{t_P}(p_1$+$p_2)(\sqrt{m}$+1)	x

The case 2 query is for limited-range, spatial-aggregate, non-filtering, and one-dimensional. In this example, we need to have $\frac{t_D}{t_P}$ factor to determine frequency, which is how many transmissions are in a given limited range of time. The total number of transmissions in the whole network is equal to the sum of all the rows. $n_S(\frac{t_D}{t_P}+1)\sqrt{n}$ is for both local storage and external storage in geographical routing. $(\frac{t_D}{t_P}+1)(n_S+1)\sqrt{n}$ is for data-centric storage. $\frac{t_D}{t_P}(n_S+\sqrt{m}-1)+\sqrt{m}+1$ is for local storage, and $(n_S\frac{t_D}{t_P}+1)(\sqrt{m}+1)$ is for external storage in hierarchical routing.

The case 3 query is for limited-range, temporal-aggregate, filtering, and one-dimensional. The factor p_i is needed for local storage and data-centric storage due to the element filtering that satisfies the temperature condition. They don't need to send all the values they measure. Among them, local storage has least number of transmissions since it aggregates and filters at the sensing node itself after t_D minutes pass. $(p_i+1)\sqrt{n}$ is for local storage, $(\frac{t_D}{t_P}+1)\sqrt{n}$ is for external storage, and $(p_i+\frac{t_D}{t_P}+2)\sqrt{n}$ is for data-centric storage in geographical routing. Similarly, $(p_i+1)(\sqrt{m}+1)$ is for local storage, and $(\frac{t_D}{t_P}+1)(\sqrt{m}+1)$ is for external storage in hierarchical routing.

The case 4 query is for continuous, spatial-aggregate, filtering, and multi-dimensional. This query has two dimensions, which are temperature and humidity. Since it has also filtering, we need to have p_1 and p_2 for probabilities of satisfying the two conditions, temperature and humidity, respectively. To calculate number of samplings, t_L is used instead of t_D, since it is continuous query. $\frac{t_L}{t_P}(n_d(n_S-1)+(p_1+p_2))\sqrt{n}+n_S\sqrt{n}$ is for local storage, $(n_d\frac{t_L}{t_P}+1)n_S\sqrt{n}$ is for external storage, and $(\frac{t_L}{t_P}(p_1+p_2+n_dn_S)+2n_S)\sqrt{n}$ is for data-centric storage in geographical routing. $(\frac{t_L}{t_P}(p_1+p_2)+n_S)(\sqrt{m}+1)+n_d\frac{t_L}{t_P}$ is for local storage and $(n_d\frac{t_L}{t_P}+1)n_S(\sqrt{m}+1)$ is for external storage in hierarchical routing. So far, all the number of transmissions is for the worst case.

4.4.2 Energy

[83] suggests the total amount of energy consumed by the network for each transmitted packet as the summation of energy consumed for transmitting, receiving, and reading only header. Based on this, we have our own model for energy consumption of unicast way in the networks and it neglects energy for processing:

$$E_u = e_{tx} + e_{rx} + (m_i - 1)e_{oh} \tag{4.1}$$

where E_u is the total amount of energy consumed by networks for transmitting one packet by unicast way, e_{tx} is the amount of energy consumed by a sender for transmitting one packet, e_{rx} is for receiving one packet, e_{oh} is for overhearing any packet that is not destined to the node, and m_i is the number of neighbors that are within a radio range of the i^{th} node. Based on this basic model, we can extend it to the amount of energy consumed by peer-to-peer unicast way.

We can derive the maximum amount of energy spent by transmitting a packet from source to sink based on the equation (4.1).

$$E_{ss} = \sum_{i=1}^{\sqrt{n}} E_u \tag{4.2}$$

and if we plug (4.1) in (4.2), we get,

$$\begin{aligned} E_{ss} &= \sum_{i=1}^{\sqrt{n}} (e_{tx} + e_{rx} + (m_i - 1)e_{oh}) && (4.3) \\ &= \sqrt{n}(e_{tx} + e_{rx}) + (7 \cdot \sqrt{n} - 6)e_{oh} && (4.4) \end{aligned}$$

When we consider the total energy dissipated by all the sensors in the network, we can analyze it by the same way we did in section 4.4.1. For example, if we consider the first case, $E_u\sqrt{n}$ is spent for sending query and the same amount is spent for sending an answer since

E_u is an amount of energy that is consumed by sending a packet from a source to a sink. Therefore, the total energy consumed by a query case 1 in local storage is $2E_u\sqrt{n}$, which is $2E_{ss}$. By the same way, we can have $2E_{ss}$ and $4E_{ss}$ for external storage and data-centric storage, respectively. Since these results are proportional to the number of transmissions in the network intuitively, we only focus on the amount of energy dissipated in a hot spot node, which can be a sink, an aggregate node, or a node for data-centric storage depending on situations. This is needed since the network is not available if a hot spot node with an important role dies. Here, we have also several assumptions: a sink is at one of the 4 corners. An aggregate node or a storage node for the data-centric storage is one of the inner nodes, whose number of neighbors are 8.

Table 4.7: Energy dissipated in a hot spot node in case 1

routing	Geo					Hier		
storage	LS		ES	DCS		LS		ES
node	sink	aggN	sink	sink	storage	sink	aggN	sink
Q	$e_{tx}+e_{oh}$	x	$e_{tx}+e_{oh}$	$e_{tx}+e_{oh}$,$e_{tx}+e_{oh}$	e_{rx}	$c_{tx}+c_{oh}$	$c_{tx}+c_{rx}$	$c_{tx}+c_{oh}$
S	x	x	e_{rx}	x	e_{rx}	x	x	c_{rx}
A	x	x	x	x	x	x	x	x
R	e_{rx}	x	x	e_{rx}	$e_{tx}+e_{oh}$	c_{rx}	$c_{tx}+c_{rx}+c_{oh}$	x

Table 4.8: Energy dissipated in a hot spot node in case 2

routing	Geo					Hier		
storage	LS		ES	DCS		LS		ES
node	sink	aggN	sink	sink	storage	sink	aggN	sink
Q	$n_S(e_{tx}+e_{oh})$	$e_{tx}+e_{rx}+7e_{oh}$	$n_S(e_{tx}+e_{oh})$	$n_S(e_{tx}+e_{oh})$,$e_{tx}+e_{oh}$	e_{rx}	$c_{tx}+c_{oh}$	$c_{tx}+c_{rx}$	$c_{tx}+c_{oh}$
S	x	x	$n_s e_{rx}$	x	$n_s \frac{t_D}{t_P} e_{rx}$	x	x	$\frac{t_D}{t_P} n_S c_{rx}$
A	x	$n_s e_{rx}$	x	x	x	x	$n_S c_{rx}$	x
R	$\frac{t_D}{t_P} e_{rx}$	$\frac{t_D}{t_P}(e_{tx}+e_{oh})$	x	$\frac{t_D}{t_P} e_{rx}$	$\frac{t_D}{t_P}(e_{tx}+e_{oh})$	$\frac{t_D}{t_P} c_{rx}$	$\frac{t_D}{t_P}(c_{tx}+c_{oh})$	x

Table 4.9: Energy dissipated in a hot spot node in case 3

routing	Geo					Hier		
storage	LS		ES	DCS		LS		ES
node	sink	aggN	sink	sink	storage	sink	aggN	sink
Q	e_{tx}+e_{oh}	e_{tx}+e_{rx}+7e_{oh}	e_{tx}+e_{oh}	e_{tx}+e_{oh},e_{tx}+e_{oh}	e_{rx}	c_{tx}+c_{oh}	c_{tx}+c_{rx}	c_{tx}+c_{oh}
S	x	x	$\frac{t_D}{t_P} e_{rx}$	x	$\frac{t_D}{t_P} e_{rx}$	x	x	$\frac{t_D}{t_P} n_S c_{rx}$
A	x	x	x	x	x	x	x	x
R	$p_i e_{rx}$	p_i(e_{tx}+e_{oh})	x	$p_i e_{rx}$	p_i(e_{tx}+e_{oh})	$p_i c_{rx}$	p_i(c_{tx}+c_{oh})	x

Table 4.10: Energy dissipated in a hot spot node in case 4 (geographical routing)

storage	LS		ES	DCS	
node	sink	aggN	sink	sink	storage
Q	n_S(e_{tx}+e_{oh})	e_{tx}+e_{rx}+7e_{oh}	n_S(e_{tx}+e_{oh})	(n_S+1)(e_{tx}+e_{oh})	e_{rx}
S	x	x	$n_d \frac{t_L}{t_P} n_S e_{rx}$	x	$n_d \frac{t_L}{t_P} n_S e_{rx}$
A	x	$n_d \frac{t_L}{t_P}$(n_S-1)e_{rx}	x	x	x
R	$\frac{t_L}{t_P}$(p_1+p_2)e_{rx}	$\frac{t_L}{t_P}$(p_1+p_2)(e_{tx}+e_{oh})	x	$\frac{t_L}{t_P}$(p_1+p_2)e_{rx}	$\frac{t_L}{t_P}$(p_1+p_2)(e_{tx}+e_{oh})

We adopt only approximate unitless ratio of 5:3:1 to give values for energy of transmitting, receiving, and overhearing from [84]. Also, we adopt channel path loss model from [51],

$$\rho = a\delta^{\gamma} + b \tag{4.5}$$

where ρ is transmission power, δ is a distance between sending node and receiving node, γ is power loss constant ($2 < \gamma < 4$), and a and b are constants. We choose $\gamma = 3$, $a = 1$, and $b = 0$ to simplify the model. Therefore, a cluster head consumes 27 times more energy than a general node in geographical routing in transmitting, receiving, and overhearing a packet. The notation for energy consumed by cluster heads is c_{tx}, c_{rx}, and c_{oh}, respectively, as shown in Table 4.7.

Table 4.11: Energy dissipated in a hot spot node in case 4 (hierarchical routing)

storage	LS		ES
node	sink	aggN	sink
Q	c_{tx}+c_{oh}	c_{tx}+c_{rx}	c_{tx}+c_{oh}
S	x	x	$n_d \frac{t_L}{t_P} n_S c_{rx}$
A	x	$n_d \frac{t_L}{t_P}$(n_S-1)c_{rx}	x
R	$\frac{t_L}{t_P}$(p_1+p_2)c_{rx}	$\frac{t_L}{t_P}$(p_1+p_2)(c_{tx}+c_{rx})	x

Table 4.7 is an example that shows energy dissipated in a specific node in terms of each storage when query case 1 is applied to. 'aggN' stands for an aggregate node, and 'storage' is a storage node that is used in data-centric storage scheme. For example, in Table 4.7, e_{tx}+e_{oh} energy is consumed to send a query at a sink node in local storage of geographical routing scheme. Here, we don't consider energy for receiving a packet from a base station. When a sink node sends a query packet, it needs to spend e_{tx}, and when its neighbor relays this packet to the third node, the sink node can overhear this packet since the transmission ranges are the same. When an answer is sent from a source node to the sink node, it spends e_{rx}, since the packet is destined to the sink node. Also, we don't consider the energy for relaying this packet to the base station. By the same way, we can get the whole energy model for the other three query cases, which is not shown due to the space. Through the summation of these four steps, we can get the whole energy consumed by a specific node, which can be a candidate for hot spot node in the network. For the local storage, a sink and an aggregate node can be candidates for hot node. A sink node and a storage node can also be candidates for hot node in a data-centric storage. Comparisons through evaluation of each value is discussed in section 4.4.5.

4.4.3 End-to-end Delay

In case of reducing energy consumption in sensor networks by decreasing the transmission range, there is a trade-off between energy and delay. If we have shorter transmission range, it will increase the number of hops for packet delivery, and it finally affects the end-to-end delay. Therefore, we need to consider end-to-end delay with energy consumption. We assume the end-to-end delay is proportional to the number of hops from a source to a sink. Since we consider only the worst case, the number of hops from a source to a sink can be bounded by linear combination of $\sqrt{n}$ or $\sqrt{m}$, which are the number of hops from the farthest source to a sink. For the local storage and external storage in geographical routing scheme, the end-to-end delay is $\sqrt{n}$, for the data-centric storage in geographical routing scheme, it is $2\sqrt{n}$,

since it needs to send query to a storage node and get the answer from it. And for both local and external storage in hierarchical routing scheme, it is $\sqrt{m}+1$. For the delay point of view, hierarchical way is superior to the other schemes since it uses cluster heads to relay packets, which can use fewer number of hops to transmit data to a sink than geographical scheme.

4.4.4 Life span and local storage capacity

If any answers from sensing nodes cannot be forwarded to a sink, the network is not any more useful. Due to the energy depletion in hot spot node or any reasons that make the network useless, we need to consider life span, how long does the network work. Meanwhile, if local storage is filled up with data, it is going to lose future data. Since we do not suggest any advanced algorithm to treat these, we just compare basic characteristics of each combination of storage and routing scheme.

4.4.5 Evaluations for Metrics

Table 4.12 shows all the results from the three metrics. Here, we used no units to compare relative values in each case. For cost evaluation, we used 9 sensor nodes for n_S, 60 minutes for t_D, 1 minute for t_P, 1 for probability of p_i, 240 minutes for average life time of sensors, which is t_L, 1 for p_1, which is probability for temperature being greater than 70, and 1 for p_2, which is probability for humidity being between 20 and 30. Since we consider the worst case, we choose 1 for all the probabilities.

Any storage types in hierarchical routing outperforms geographical routing in terms of the number of transmissions. But, as we mentioned earlier, hot nodes will die faster than nodes in geographical routing since the cluster heads consume more energy due to the longer transmission range. In case 3, local storage outperforms other schemes in terms of both the number of transmissions and energy since case 3 contains temporal aggregate query. For the temporal aggregate, the sensing node does not need to send data to an aggregate node, since the

node itself aggregates all the data, which saves energy. For the filtering query such as case 4, local storage and data-centric storage can reduce the number of transmissions and save energy if the probability that satisfies filtering condition decreases. When we give 0.1 as p_1 and p_2, the number of transmissions for local storage and data-centric storage drop from 428,571 and 476,190 to 385,803 and 433,422, respectively.

Figure 4.2 shows the comparative analysis with different weights for the number of transmissions, energy, and delay. Figure 4.2(a) has equal weights, while each of figures 4.2(b)$\sim$4.2(d) give extra weight to one of the three measures. The values in Table 4.12 are normalized to the biggest number in each metric. These values are calculated to get normalized weighted cost.

For Figure 4.2(a), three metrics have equal weights as $1/3$. In this case, external storage in hierarchical routing has the best cost for query case 1 and local storage in hierarchical routing has the best cost for query cases 2 to 4. This result is applied to figures 4.2(b) and 4.2(d), which are weighted on the number of transmissions and end-to-end delay. Meanwhile, for Figure 4.2(c), which is weighted on the metric energy, local storage in geographic routing has the best cost for query cases 1, 3, and 4, external storage in geographic routing has the best cost for query cases 1 and 2. This comparative analysis shows what kind of storage has the best cost, when a specific metric is emphasized.

4.5 Experimental Results

In this section, we evaluate three metrics by simulation to have average cases.

4.5.1 Assumptions for simulation

The link status is robust, so there is no retransmission due to the channel error. Nodes are static and distributed evenly on the square grid. The number of nodes is 9801, which

is same as in the section 4.4. Nodes in geographical routing are homogeneous but ones in hierarchical routing are not homogeneous. A cluster head has transmission range that can cover its eight cluster head neighbors, but a cluster member can only cover its one-hop neighbors. The fundamentals in simulation is based on cost analysis, which are introduced in Table 4.4 through Table 4.11. Discrete time event driven simulator is used to simulate all the combinations of four query cases and five different routing and storage types.

For a geographical routing, a sink node is generated randomly among the nodes in the right most side (column), and a source is generated randomly at any nodes except a sink node itself and nodes in four sides. And an aggregate query such as case 2 and 3 restrict nodes to be aggregated as neighbors for the source, which are generally 9 nodes including the source node itself. For a hierarchical routing, a sink node is generated randomly among the right most cluster heads, and a source can be any node regardless of its being cluster head or member. But for spatial aggregate query such as query case 2 and 4, a query is restricted to have 9 nodes to be aggregated.

In addition to counting the number of transmission at each node, the number of receiving a packet and the number of overhearing a packet are counted to calculate energy consumed in the whole network or a specific node, which can be a hot-spot node. Therefore the energy in Table 4.12 represents a hot-spot node in the network, which can be either a sink node or aggregation node. A delay is measured from a query packet is sent into the network to an answer packet is received at the sink node. But for the query cases 2,3, and 4, which has periodic sensing duration, the delay is summation of the time for a query packet being delivered to the source node and the time for the last answer packet being sent to the sink node. The simulation ends when the last packet is sent back to the sink or when a certain node runs out of energy, which causes the network not to be alive.

4.5.2 Results for the simulation

Table 4.12 shows the simulation results for the three metrics depending on 4 query cases and 5 storages. Figure 4.2 shows the comparative analysis with different weights for the number of transmissions, energy, and delay in average case. For the equal weights on the three metrics (figure (4.2(a))), hierarchical local storage is most efficient overall in four different query cases when three metrics are equally considered. Hierarchical local storage has the overall lowest number of transmissions and the overall delay since it uses longer transmission range that shorten the number of hops for forwarding packets. The merit in the number of transmissions and the delay bring the overall best cost to the hierarchical local storage in Figure 4.2(a). But hierarchical storages are not efficient to be used in one-shot and non-aggregate query such as query case 1 since the geographical storages outperform them.

External storages for both geographical and hierarchical routing is not suitable for limited range or continuous query cases since they have to send data periodically without having any aggregation or filtering, which increases the number of transmissions and energy consumption. In the delay point of view, geographical data centric storage has the worst cost, since it needs extra hops to save sensed data and retrieve them.

Table 4.12: Results for the simulation

metrics	Number of transmissions					Energy					Delay				
routing	Geo			Hier		Geo			Hier		Geo			Hier	
storage	LS	ES	DCS	LS	ES	LS	ES	DCS	LS	ES	LS	ES	DCS	LS	ES
case1	116.4	116.4	224.9	41.8	41.8	0.1	0.1	0.2	3.6	3.6	1.2	1.3	1.8	0.5	0.5
case2	1185.8	5761.8	4242.6	303.2	1919.2	3.2	8.2	7.5	25.1	219.5	1.4	1.5	1.9	0.7	0.7
case3	94.5	640.2	512.2	31.6	209.0	0.1	0.9	1.0	3.3	24.9	1.1	1.4	1.1	0.4	0.5
case4	1467.3	10999.8	7337.2	450.9	2425.5	5.8	16.3	14.8	32.4	300.3	1.4	1.6	1.9	0.7	0.4

4.6 Summary

We classify sensor network queries and apply them to various storage types that are distinguished by different routing schemes. We have five different criteria and several sub-classification for queries. Besides evaluations for query types versus storage types based on fixed parameters, we can verify the results by simulation. This is needed since we only evaluated the worst case.

In this chapter, we re-establish the classification of queries in sensor networks, data delivery models, storage types, and data dissemination schemes. After we pick four representative queries, we apply them to storage types by means of applying different kinds of data dissemination schemes. We see the difference of maximum possible number of transmissions in its evaluation depending on the combinations of query types and storage types. When a query has specific characteristics, it has a storage type that match well the query in terms of the number of transmissions, energy, delay, or some combinations of them.

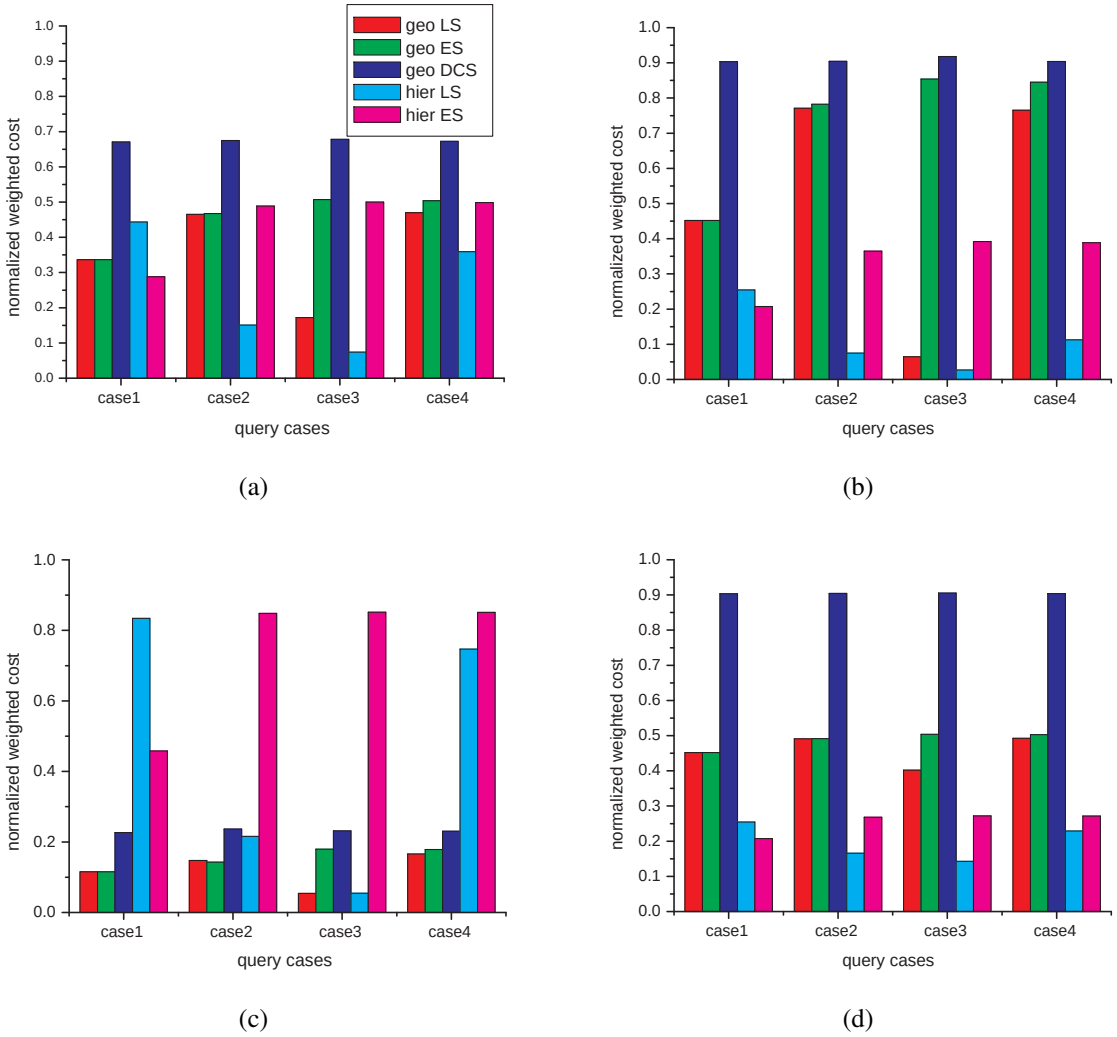


Figure 4.2: Performance cost comparison for each query case and storage type with different weight for the three metrics (a) Equal weights on the three metrics (b) Weight on the number of transmission (c) Weight on energy (d) Weight on end-to-end delay

CHAPTER 5

ENERGY-EFFICIENT INDEXING

5.1 Introduction

Among the many possible query types and network schemes in the previous chapter, we focus on spatial query and network scheme that uses geographical tree routing, since hierarchical network scheme generally consumes more energy than plain type network scheme. In this chapter, we propose a new network scheme for energy-efficient query dissemination. This scheme has many virtual squares that have locally organized trees so that a query can be disseminated into the particular square area without consuming extra energy. We add this scheme on MBR indexing so that we can save more energy than a network scheme that only uses MBR indexing. This network scheme is called sectioned tree. We then simulate our sectioned tree in different conditions and measure energy consumption for two different queries.

The remainder of this chapter organized as follows. Section 5.2 presents motivation for this work. We describe the sectioned tree construction, how to save energy and cost evaluation in section 5.3. Section 5.4 shows our experimental result, when sample queries are applied to the sectioned tree. Finally, we summarize this chapter in section 5.5.

5.2 Motivation

5.2.1 Energy-efficient network model

In order to disseminate queries into sensor nodes and gather appropriate data from them, it is required to have energy-efficient network structure since the sensor network is energy limited. Because of this, many researchers have tried to find optimal energy-efficient network structure to increase network lifespan, but they find that an optimal solution to minimize en-

ergy consumption in wireless sensor networks or wireless networks that use broadcast is an NP-complete problem [41, 58]. Sensor network topology can be mapped to a graph, which has sensor nodes as vertices and links between nodes as edges. From this, a routing tree for fully aggregated queries [1] with reception cost is proven to be NP-complete, which is the same problem as minimum degree spanning tree (MDST) that is known to be NP-complete [41]. Also, it is necessary to save energy when a sensor node forwards a query message. In this case, usually broadcasting method is used and minimum energy broadcasting data problem in two dimensional Euclidean metric space is also proven to be NP-complete [58]. Our focus is to find an energy efficient network structure that saves energy when queries are disseminated and data is sent back to a base station by proposing the sectioned indexing tree for spatial queries.

5.2.2 Energy model

Our energy model is based on [38], which considers energy consumption in overhearing message. Overhearing energy is consumed when a node receives a message that is not destined to it due to being within the radio range of a sender. Energy consumed in transmission (E_{tx}) is proportional to the power of 2 to 4 due to path loss depending on the medium.

$$E_{tx} = E_{txelec} + \epsilon d^{\alpha} \tag{5.1}$$

where, E_{txelec} is the energy consumption by transmitter electronics, ϵ is an amplifier characteristic constant, d is a distance between two communicating nodes, and α is the path loss, which is 2 in this paper. Energy consumed in reception is independent of the distance and it needs only E_{rxelec}, which is generally same as E_{txelec}, and we define E_{elec},

$$E_{elec} = E_{txelec} = E_{rxelec} \tag{5.2}$$

[1] This produces one result value from several nodes that are related to the query

For overhearing, we consider the node consumes less energy than that of receiving data, since it is possible for a sensor node to have a mechanism of early discarding of a message that is not destined to it [38]. Therefore, we assume the ratio of energy spent in receiving and overhearing data is 1:β (,where $0 < \beta < 1$).

$$E_{oh} = \beta E_{elec} \tag{5.3}$$

When a message is transmitted from a node u to v, the amount of energy consumed by nodes that are participating the transmission, E_{uv}, is the sum of energy of transmission, reception, and overhearing.

$$\begin{aligned} E_{uv} &= E_{tx} + E_{rx} + (n-1)E_{oh} \\ &= (2 + \beta(n-1))E_{elec} + \epsilon d^{\alpha} \end{aligned} \tag{5.4}$$

where, n is the number of neighbors that is within the radio range of u. In equation 5.4, since β, E_{elec}, ϵ, and α are constants, and n is dependent on d, so d is the dominant factor for E_{uv}. Also, since the per byte energy consumed in communication is a couple of thousand times higher than in any other internal processing, such as cpu execution and memory handling [27], we assume that we disregard the amount consumed by running Dijkstra's or Prim's algorithms or any internal query or data processing.

5.2.3 Spatial queries

Queries in sensor network have been classified using different criteria [29, 33, 32, 28, 85, 30]. In any sensor network applications that require spatial and temporal information, such as monitoring applications, queries can be classified based on point and range of space and time. A point in spatial query represents a location of one sensor node in which a query is interested. A range in a spatial query represents an area, which could be a rectangle that includes one

or more sensor nodes. A point in temporal query means a specific moment. And, a range in temporal query means a time interval that has a logical starting point and a logical end point. Among these, we focus on a space range query, whose packets are disseminated into a certain region and sensor nodes in the region should return in-network aggregated value that represents the region.

5.2.4 In-network data aggregation and MBR indexing

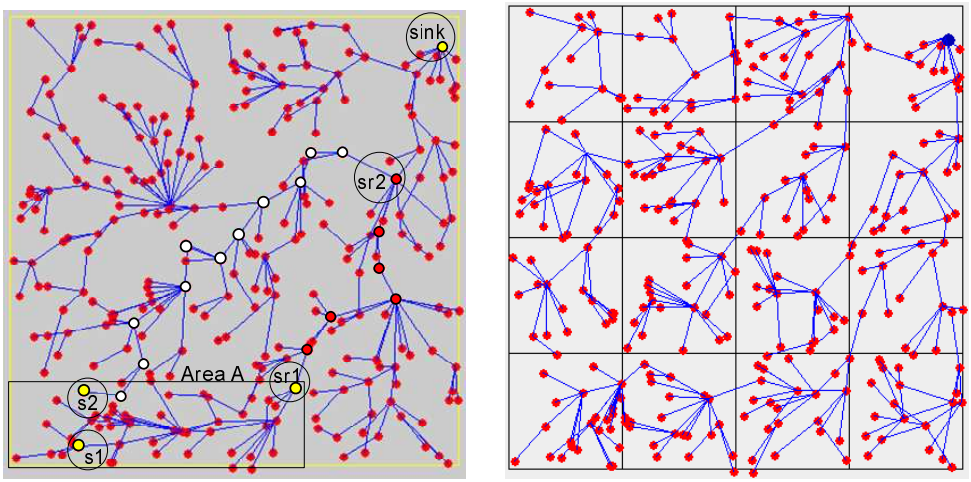
Traditional in-network data aggregation methods such as TAG [28], are not suitable for a spatial query that has location information, since it supports one dimensional attribute conceptually but it does not consider physical location of nodes, which can be represented as two dimensional attributes. Due to this reason, [54] suggests a distributed spatial index (SPIX) for a spatial query on top of a routing tree. In SPIX, every node has its MBR that includes itself and all of its descendants. A node can transmit queries only to any of its children who has an MBR that intersects the query MBR.

In-network data aggregation works as follows: when data is gathered and aggregated in a network, leaf nodes can just pass data up to their parents. A parent waits until all the children report their data and aggregates the data based on the given query. This process is repeated recursively until the final aggregate is calculated at the root node.

5.3 Sectioned Tree

5.3.1 Overview of sectioned tree

We assume each node has a global positioning system (GPS) and that nodes are randomly deployed in a square area. The sequence of building a sectioned tree has three phases. In phase I, global information, such as MBR of the entire network, the number of sections, and positions of a sink node and anchor of each section, is exchanged globally. And local information is exchanged within each section. In the phase II, local trees that are determined by Dijkstra's



(a) (b)

Figure 5.1: Two tree models (a) Naive routing tree (b) Sectioned tree

shortest path algorithm or Prim's minimum spanning tree algorithm are constructed at each section. In phase III, local trees are interconnected each other and make one global tree. An MBR for spatial indexing is assigned to each node in the network during phase III.

5.3.2 Construction of sectioned tree

1. Initial information exchange : First, a base station sends information, which divides sensor network area into several sections, to all the nodes by flooding. The shape of each section and the number of sections are arbitrary, but in our assumption the shape is square and the number of sections divide the network in a grid of equally sized sections. Initial section information looks as in Fig. 5.2:

Position of a base station (x,y)	MBR of entire network (x1,y1,x2,y2)	The number of sections	Anchor position for sub-roots (x,y)

Figure 5.2: Global information

where, the last column is to determine a sub-root among all the nodes, which is closest to that anchor position. Second, all the sensor nodes in the network identify their positions and calculate sections and ID of the sections they belong to. Then they exchange their information by transmitting message only to their one-hop neighbors, as illustrated in Fig. 5.3:

Node ID	Node position (x,y)	Section ID	Section MBR (x1,y1,x2,y2)	Sub-root position (x,y)

Figure 5.3: Local information

Third, they create neighbor list and add the record of Fig. 5.3 whenever the node receives information. By now, all the nodes can have base station information (position), local information as in Fig. 5.3, neighbor list with neighbor ID, neighbor position, and section ID where the neighbor belongs to, and its section information.

2. Construction of local trees : First, in each section, nodes calculate the distance to the anchor point from the neighbors and themselves. And if the closest node to the anchor point is the node itself, that node is designated as a local sub-root in the section. In this case, there is no communication between nodes. Second, after a certain amount of time, sub-root nodes broadcast their information to every other node in the same section to build initial local temporary trees. In this case, distributed tree building method is used that broadcasts tree building message to be forwarded recursively until all the leaf nodes get the message and set their levels. After a certain amount of time, leaf nodes will start their back tracking up to the local sub-roots. Now the sub-root nodes can know the whole picture of its section. Third, every local sub-root nodes can run either Dijkstra's shortest path algorithm or Prim's minimum spanning tree (MST) to build their energy efficient local trees by the locally centralized method. These two known algorithms eventually set a tree when we give a specific node as a root. Since getting an optimal solution for energy efficient topology is NP-complete, we show that the

trees made by Dijkstra's and Prim's algorithm outperforms the naive tree in terms of energy consumption when a message is delivered from a node to the other. Fig. 5.4 shows our pre-experiment that compares three kinds of trees at different number of nodes, 50, 100, 200, 500, and 1000.

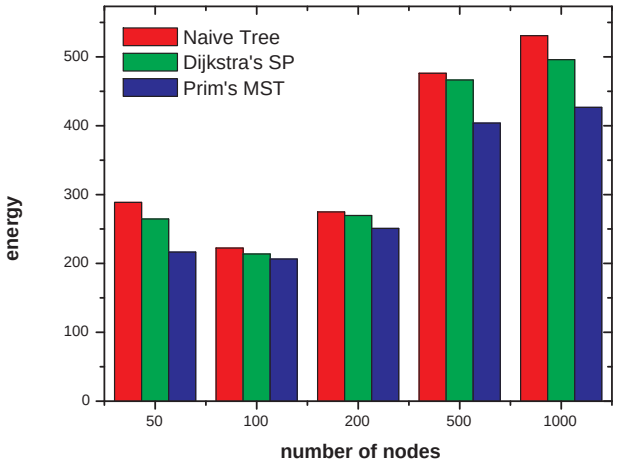


Figure 5.4: Energy consumption in different tree structure

When we use Dijkstra's algorithm or Prim's MST, the global information is needed to calculate them. If this is applied to wireless sensor network, one of the problems is to transmit the calculated tree structure to all the nodes in the network. But this is not scalable nor energy efficient since it needs to flood the global topology to all the nodes, but the payload size of current technology in sensor network is limited. Therefore, one packet can not contain all the global information of tree structure and we need at least more than one packet, which makes several number of flooding, which causes eventually network-wide energy consumption. Although Dijkstra's algorithm or Prim's MST consumes energy a lot when the information is transmitted, we can alleviate the load of information transmission or the load of calculation at all the nodes in the network, since our proposed scheme runs these algorithms at local sub-

roots, where they can take care of certain number of nodes, which is determined by the number of nodes and the number of sections in the network. Also, this is scalable as the number of nodes increases, since we can give more number of sections that can bound the number of nodes in one section.

The difference of properties produced by the two algorithms is as follows. Dijkstra's algorithm looks for the shortest path from a root to every node in a network. Since we give Euclidean distance on the link as a weight of Dijkstra's algorithm, initial radio range of nodes can be dominant to determining the shape of the tree. On the other hand, if we use Prim's algorithm with the Euclidean distance again on the link as a weight, the initial radio range of sensor nodes does not affect the shape of a tree, since the Prim's algorithm always looks for the nearest node that is not in the current spanning tree. Therefore, as a result of these properties, energy consumption or the number of hops from a sink to a source can vary. We have various cases of initial radio range so as to see the different results in our simulation.

3. Local trees interconnection : Now we need to build one big tree so that a message can be disseminated to any node in the network. First, sub-root nodes exchange messages with inter-section nodes, which are outside the section but within its radio range, and list them. Then, later on, local sub-root nodes can use one of the inter-section neighbors as a link to connect local trees to each other. The pseudocode for determining interconnection node is shown in Fig. 5.5. Basically, a sub-root node tries to find a neighbor that satisfies the following condition. A neighbor is in the other section, and an angle made by position of the sub-root, the neighbor, and the base station is less than or equal to $\pm 30^{\circ}$. And the sub-root choose the neighbor, whose angle is smallest. If the sub-root cannot find an appropriate node, it increases its radio range up to its maximum power and build the neighbor list again. Then, it repeats the angle calculation and determine its interconnection node. In Fig. 5.5, s_n in line 5 means section ID of neighbor and rr_{sr} in line 10 means radio range of the sub-root. Second, after a certain amount of time, the base station sends a message for building the global tree. Third,

leaf nodes send MBR back to the base station. Along the path from leaf nodes to the base station, in-network aggregation would be used. After this, the whole tree is created. Fig. 5.6 summarizes the sequence of construction of sectioned tree.

Algorithm 1: determining interconncetion node

Input: section s_i, sub-root *sr*, neighbor list of *rs*,
position of a base station *bs*
Output: interconnection node n_i

```
 1: for each section s_i do
 2:    sr count the number of neighbors;
 3:    if number of neighbors > 0 then
 4:       for each neighbors n do
 5:          if s_i ≠ s_n and |angle(n,sr,bs)| ≤ 30° then
 6:             let n_i = n, whose angle is smallest;
 7:             return n_i;
 8:          end if
 9:    else
10:       rr_sr = max_radio_range(sr);
11:       rebuild neighbor list;
12:       sr count the number of neighbors;
13:       if number of neighbors > 0 then
14:          for each neighbors n do
15:             if s_i ≠ s_n and |angle(n,sr,bs)| ≤ 30° then
16:                let n_i = n, whose angle is smallest;
17:                return n_i;
18:             else if n_i = Ø
19:                report isolated section;
20:             end if
21:       end if
22:    end if
```

Figure 5.5: Pseudocode for determining interconnection node

5.3.3 Cost evaluation of sectioned tree construction

Even though the construction procedure is quite complicated and costs a lot in terms of energy consumption, this cost for the construction can be recovered after a number of query dissemination and data gathering. The energy consumption is dependent on distance and number of transmission, as we can see in equations 6.1 and 5.4. If n is the number of sensor nodes in the network and m is the number of sections, the number of transmissions in the whole network can be counted at each step. In phase I, line 2 and 5 in Fig. 5.6 makes n transmissions each, since flooding and exchange needs one transmission per node. In phase II, line 9 and 12 need n transmissions each due to the same reason. In phase III, line 16 needs $m + m \cdot n_n$ transmissions, where n_n is the average number of neighbors of sub-root nodes. And, line 18 and 21 needs n transmissions each due to the same reason. So far, the cost for building a sectioned tree is $6n + m \cdot (1 + n_n)$, which is upper bounded by $7n$ since $m + m \cdot n_n$ is less than or equal to n intuitively. Therefore, asymptotic cost for tree building is $O(n)$.

5.3.4 Energy saving in query dissemination and data gathering

We build a local tree within the boundary of a given section. Thus the shape of tree or type of tree does not matter.

MBR indexing and local trees help the sensor network save energy. Saving energy by MBR indexing is intuitive since it can prevent the network from flooding the query and only forwards the query to the relevant nodes whose descendants' MBR intersect the query MBR. This can reduce the number of transmissions and eventually save energy. But why does the local tree in a section save energy?

MBR indexing itself is not very energy efficient when a query region includes two or more branches from a node that has a common nearest ancestor. For example, in Fig. 5.1(a) a query injected to the network by the sink node is forwarded to two branches by a sub-root 2 ($sr2$) toward the query area, Area A, which is a rectangle at the bottom of Fig. 5.1(a). Most

of the nodes including source 1 ($s1$) can receive the query through a sub-root 1 ($sr1$), but the source 2 ($s2$) should receive the query via white nodes along with the path between $s2$ and $sr2$. These white nodes consume extra energy that may be prevented, if the $s2$ could have followed the route that is connected to the $sr1$. The same thing might happen when data is sent back up to the sink node by the TAG-like in-network aggregation method [28]. Therefore, we suggest a sectioned tree structure for spatial indexing, which has local sub-trees in sections that divide the sensor network area, so that spatial queries be disseminated to a particular query section rather than to the whole network.

We also use adjustable transmission power and range as in [86]. After local trees are constructed by either Dijkstra's or Prim's algorithm, nodes can adjust their transmission power so that their radio range can cover only their parents and children. This can reduce the number of nodes that overhear messages from a sender and therefore saves energy.

5.4 Experimental results

In this section, we present our simulation results based on randomly deployed sensor nodes. The simulation is done in JAVA. We have 200 x 200 m^2 area with 100 nodes (0.0025 nodes/m^2) for sparse case and 400 x 400 m^2 with 1000 nodes (0.00625 nodes/m^2) for dense case. Initial radio range for all the nodes is 50 m, but they can adjust their radio range when it is requested. Sensors are all homogeneous and a sink is designated to a node that is closest to the top right corner of the network area. We have 16 different topologies that have randomly assigned source node for spatial point query or sections for spatial range query. Four types of queries include query 1 through 4, which are space point and time point (query 1), space point and time range (query 2), space range and time point (query 3), and space range and time range (query 4). Four different initial radio ranges are applied to both sparse and dense cases, and

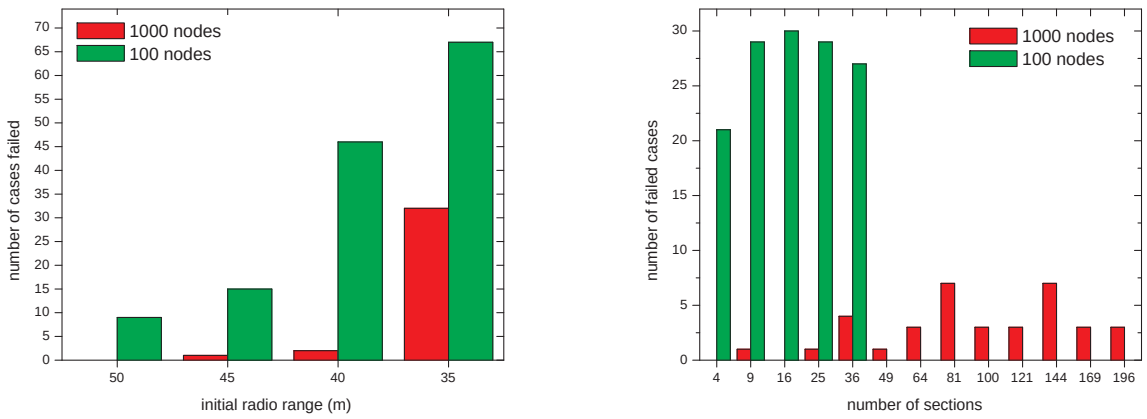
1: **Phase I** ***Initial information exchange***
2: a base station floods 'global information'
3: every nodes identify their position and determine their section
4: they set their 'local information'
5: they exchange their 'local information' with one-hop neighbors
6: they create neighbor list
7: **Phase II** ***Construction of temporary local trees***
8: all the nodes in each section determine their local sub-root
9: local sub-roots flood temporary tree building message within their sections
10: leaf nodes start to answer back to their parents
11: nodes that receive answers write down their children
12: they pass the answers up until local sub-roots receive them
13: every local sub-roots run either Dijkstra's or Prim's algorithm
14: they broadcast a local tree topology to their descendants
15: **Phase III** ***Interconnection of local trees***
16: sub-roots exchange message with its inter-section nodes
17: they determine the link connecting to other sections
18: a base station sends a global tree building messsage
19: global leaf nodes send MBR information to their parents
20: nodes that receive MBR calculate their MBR to include descendant's MBR
21: they pass the MBR up until the base station receive it

Figure 5.6: High-level description for constructing sectioned tree

they are 50, 45, 40, and 35 m. For sectioned tree, the number of sections are from 4 to 196 that are integer square from 2 to 14 for dense case and from 4 to 36 for sparse case.

When sensor nodes are deployed in the field, it is possible for some sensor nodes to be isolated depending on radio range of nodes. Initially, we count isolated cases whose data is not included in our final result. Fig. 5.7(a) and 5.7(b) shows the number of failed cases because of node isolation by different radio range or number of sections. Failed cases include node isolation, fail to interconnection, and any case that makes a tree incomplete. The number of failed cases increase as the initial radio range decreases. We show next several basic features for the simulation, which are the average of adjusted radio range of all the nodes, the average number of neighbors of each node, and the average number of hops when a source (or section) is given by a query. In Fig. 5.8, a legend with SD represents sectioned tree with Dijkstra's

algorithm in building local tree, SP represents sectioned tree with Prim's algorithm. 1000 represents the number of nodes, which is dense case, and 100 represents sparse case. Since Prim's algorithm always tries to span a tree with an edge whose weight is smallest, which is a distance in this case, the adjusted radio range is always small regardless of initial radio range (Fig. 5.8(a)). But in Fig. 5.8(b), as the number of sections increase, the adjusted radio range is slightly increasing since we use maximum radio range at sub-root nodes when they fail to find their interconnection at first. This affects both Fig. 5.8(d) and 5.8(f) since as the radio range increases, the number of neighbors within the radio range increases and the number of hops decreases. Meanwhile, Dijkstra's algorithm has an opposite trend to Prim's case. As we can see in Fig. 5.8(b), 5.8(d), and 5.8(f), as the number of sections increases, both adjusted radio range and the number of neighbors decreases, but the number of hops increases.



(a) (b)

Figure 5.7: Cases of isolated node (a) failed cases in terms of initial radio range (b) failed cases in terms of number of sections

We measure total amount of energy consumed in the whole network from queries that are disseminated into the network until the data at the nodes are sent back to the base station

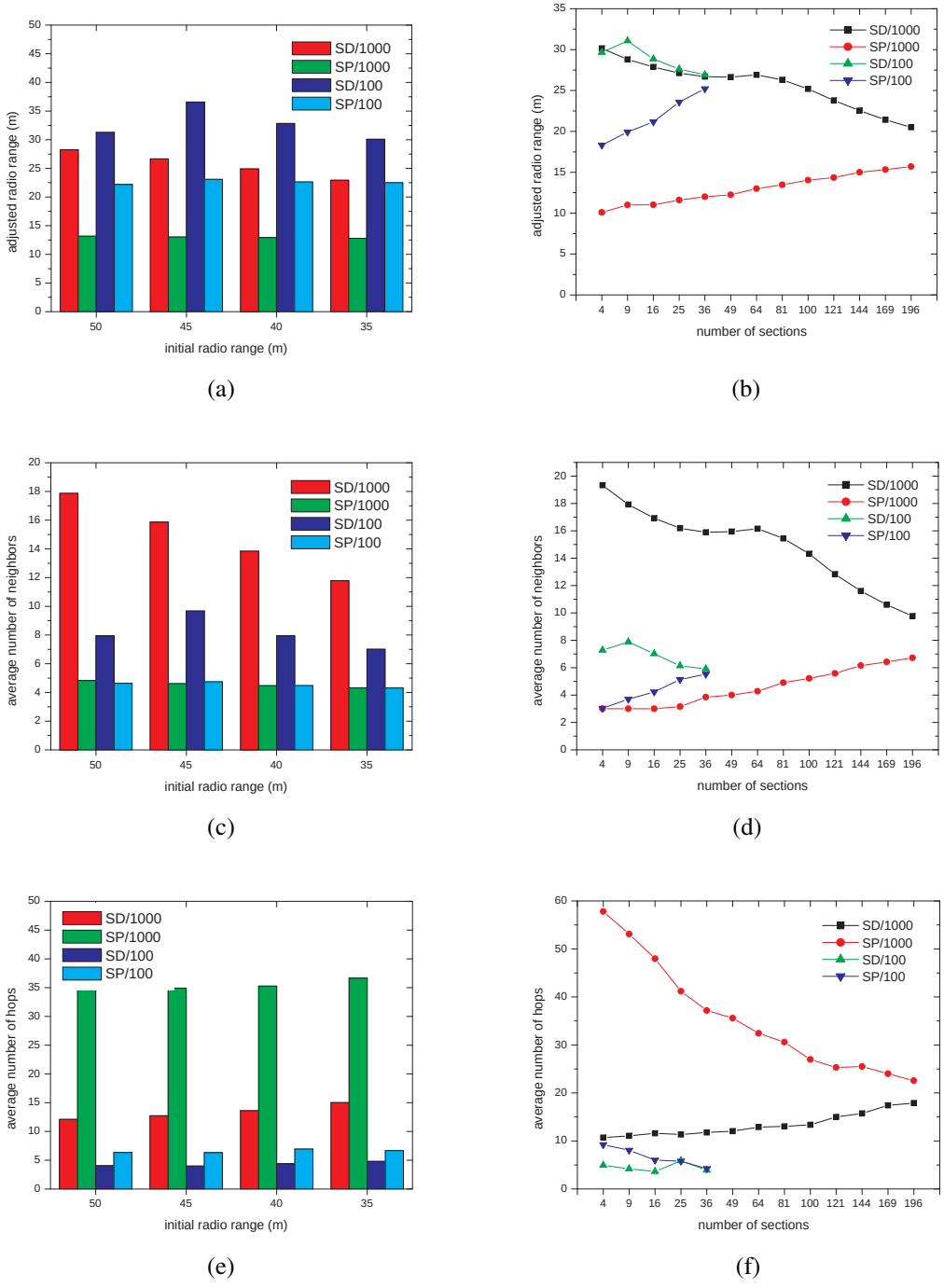


Figure 5.8: Miscellaneous information about the simulation (a) adjusted radio range in terms of initial radio range (b) adjusted radio range in terms of number of sections (c) number of neighbors in terms of initial radio range (d) number of neighbors in terms of number of sections (e) number of hops in terms of initial radio range (f) number of hops in terms of number of sections

in case of query 3 and query 4 [2]. In Fig. 5.9, SFD and SFP represent sectioned tree with flooding (without MBR indexing) of Dijkstra's and Prim's algorithm respectively. SMD and SMP represent sectioned tree with MBR indexing of Dijkstra's and Prim's algorithm, respectively. NSF and NSM represent non-sectioned tree that has one big tree in the whole network without having either Dijkstra's or Prim's algorithm. In this case, we use naive tree building algorithm. NSF means non-sectioned tree with flooding, and NSM means non-sectioned tree with MBR indexing. Since NSF uses flooding to disseminate the query, it has worst energy efficiency regardless of initial radio range. Even though NSM uses MBR indexing, which prevents nodes from flooding queries, since it does not consider minimum number of nodes to be related to a path from a sink to the query area, its energy efficiency is worse than the various kinds of sectioned tree, which are SFD, SFP, SMD, and SMP. NSF and NSM are not shown in Fig. 5.9(c) and 5.9(d), since they do not have sections. SMP consumes the least energy in both cases, query 3 and 4, regardless of the number of sections (Fig. 5.9(c) and 5.9(d)). SMD, which uses MBR indexing is the second least when the number of sections is large.

Now, we consider the energy consumption including cost for initial tree construction. The energy of building a sectioned tree is higher than that of building a non sectioned tree, since sectioned tree needs extra flooding to exchange the information for construction as we show the cost analysis in section 5.3.3. This is shown in Fig. 5.10 as a form of energy consumption. A set of sectioned tree (SFD,SFP,SMD, and SMP) is worse than non sectioned tree (NSF and NSM) in terms of energy consumption in query 3. This is because the amount of construction for sectioned tree is bigger than that of query dissemination and data gathering. But, as we give more queries and get time range result such as periodic data, then the amount of total energy consumption of sectioned tree gets better than that of non-sectioned tree. This is straightforward since non-sectioned tree consumes more energy for query dissemination and data gathering, therefore as the number of queries and data gets bigger, the total energy con-

[2]Due to space limitation, we do not show the results for query 1 and 2

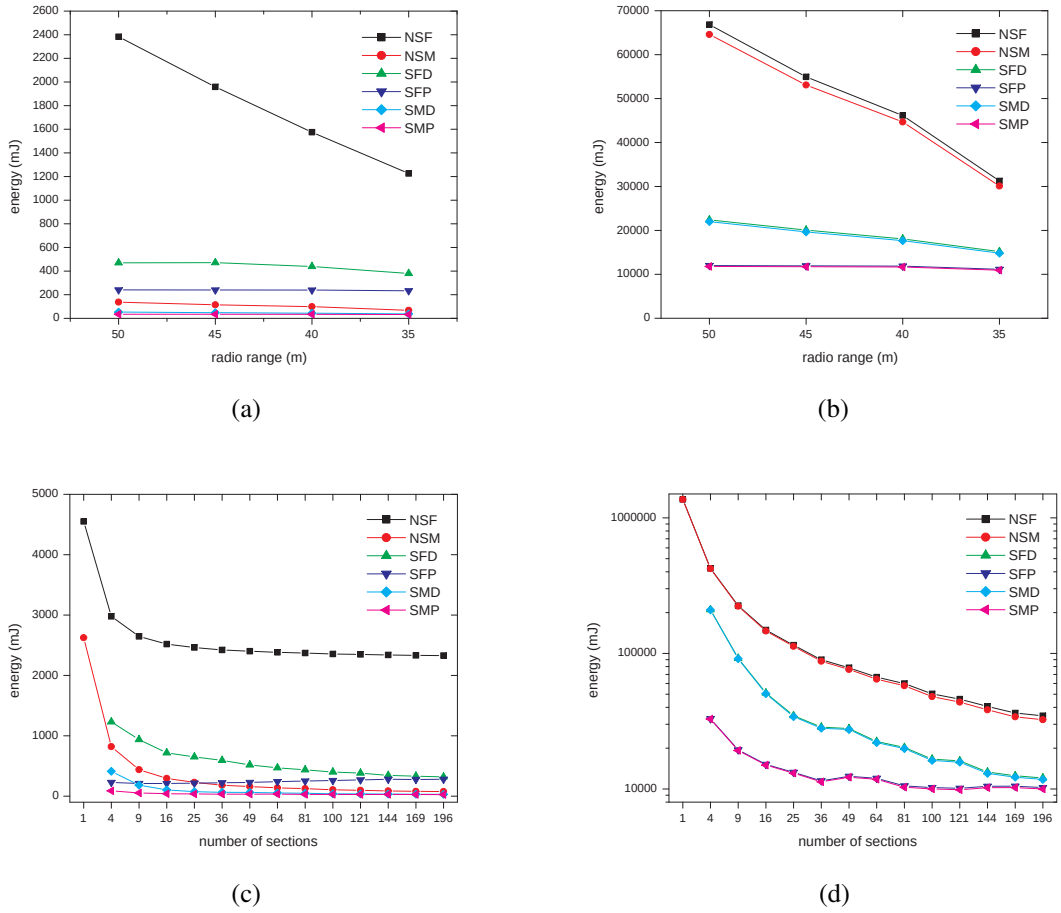


Figure 5.9: Energy consumption for query dissemination and data gathering in dense case (a) query 3 in terms of radio range (b) query 4 in terms of radio range (c) query 3 in terms of number of sections (d) query 4 in terms of number of sections

sumption of non-sectioned tree passes that of sectioned tree. Generally, schemes using Prim's algorithm have better energy efficiency than the others, since Prim's algorithm makes nodes have short radio range, which affect saving energy by decreasing the distance factor, d, in equation (5.4)

Finally, we have results for sparse case, which has 100 nodes in the network. As we can see in Fig. 5.7, sparse network has more chances for nodes to be isolated. This makes the reliability of data lower due to not having enough number of samples. But the behavior

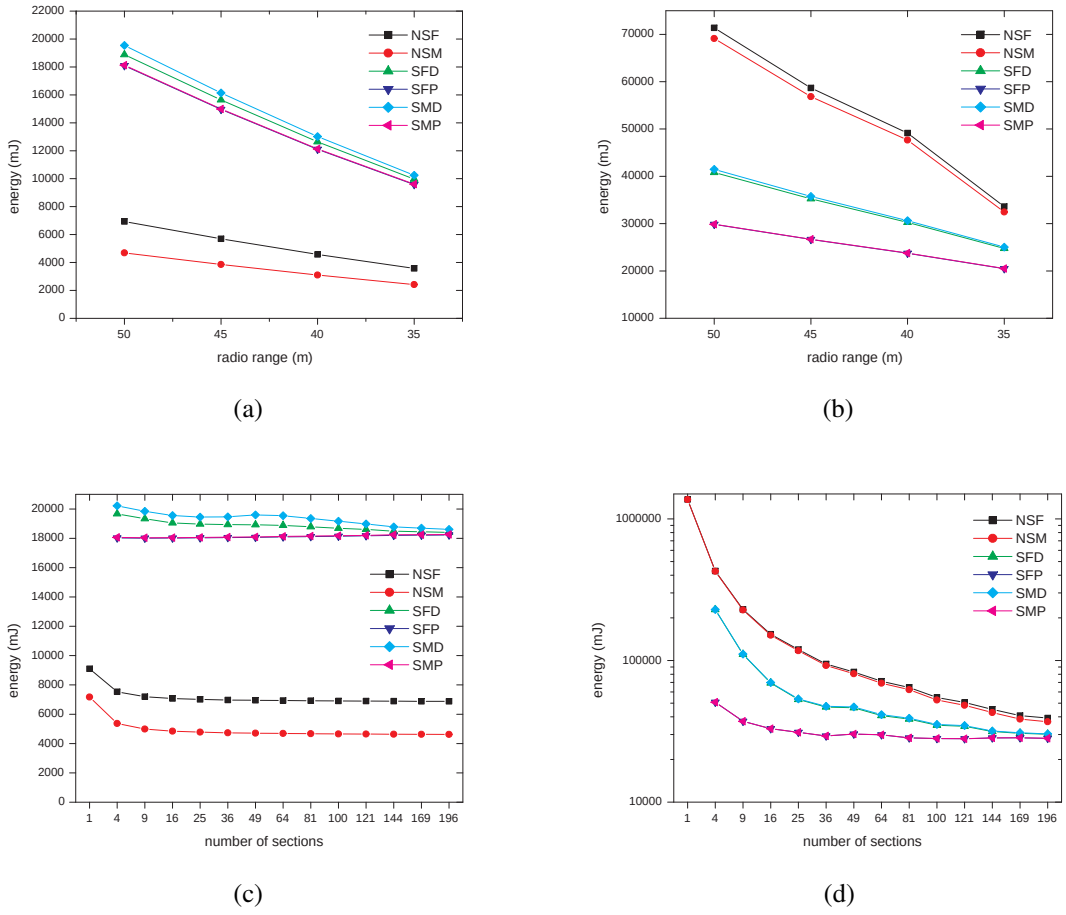


Figure 5.10: Energy consumption for query dissemination and data gathering plus initial tree construction in dense case (a) query 3 in terms of radio range (b) query 4 in terms of radio range (c) query 3 in terms of number of sections (d) query 4 in terms of number of sections

of energy consumption for both query dissemination and data gathering and total energy that include initial tree construction is similar to that in the dense case. We show only query 4 case in Fig. 5.11. Fig. 5.11(a) and 5.11(b) represent the energy consumption only for query dissemination and data gathering by different radio range and different number of sections, respectively. Fig. 5.11(c) and 5.11(d) represent total energy consumption including tree building cost by different radio range and different number of sections, respectively.

5.5 Summary

In this chapter, we suggested the sectioned tree, which has virtual sections in the network and each section has a local tree that is built within the section. This sectioned tree is energy efficient when spatial query is disseminated to an area and data is reported to a base station.

In the next chapter we extend our local tree algorithm so that it can have better energy efficiency of data gathering than known algorithms, Dijkstra's or Prim's algorithms.

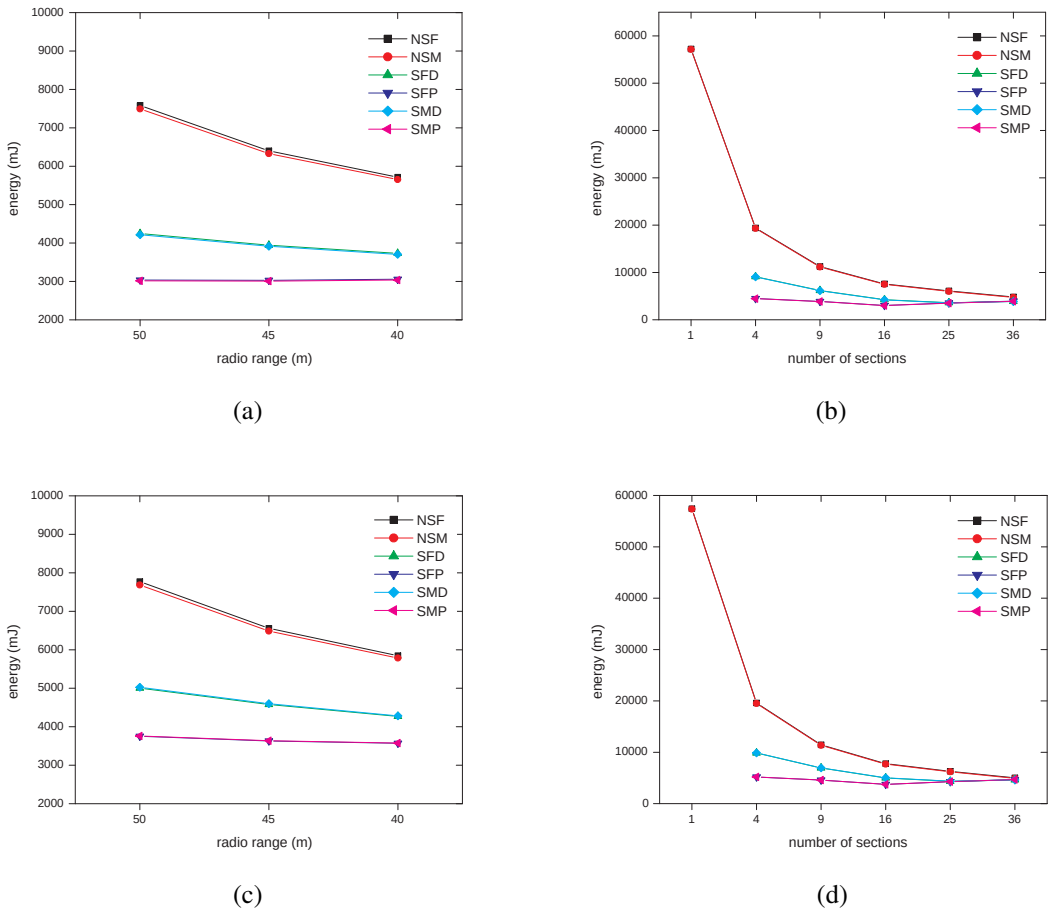


Figure 5.11: Energy consumption for query dissemination and data gathering plus initial tree construction in sparse case (a) energy for only query and data in terms of radio range (b) energy for only query and data in terms of number of sections (c) added energy for tree building in terms of radio range (d) added energy for tree building in terms of number of sections

CHAPTER 6

ENERGY-EFFICIENT DATA GATHERING

6.1 Introduction

In the previous chapter, we studied *sectioned tree* to disseminate queries in a particular region energy efficiently. In wireless sensor networks, once spatial and temporal aggregate queries that are interested in an aggregate value of a particular region are disseminated, all the nodes in the area report the result to a base station periodically. Since nodes are generally randomly deployed in the region, building an energy-efficient data gathering structure is important to keep the nodes alive as long as possible. In this data gathering tree, the direction of data flow is opposite to the one in query dissemination in the previous chapter.

We define the problem as Energy-efficient Data Gathering (EDG) tree problem and prove the problem is NP-complete. In our energy model, we consider two-way communication including transmission, reception, overhearing, and idle energy. We propose two heuristic algorithms to solve the EDG tree problem and compare the energy efficiency with other techniques.

The remainder of this chapter is organized as follows. Section 6.2 describes network and energy model that is used through this chapter. We define our EDG problem in section 6.3. We propose heuristic algorithms to solve our EDG problem in section 6.4. Section 6.5 shows our experimental results, which compare our heuristic algorithms with existing ones while varying the parameters of the energy equations. Finally, we conclude our work and discuss future work in section 6.6.

6.2 Backgrounds

6.2.1 Network Model

We first define the terms that will be used throughout this paper in a graph theoretical manner and then we describe the EDG-tree problem.

An initial undirected graph $G = (V, E)$ is given. V is a set of nodes. V has exactly one root node r and many general nodes i as its elements. E is a set of edges, whose element (i, j) is formed when node j is within an initial radio range r_{max} of i, by symmetry, node i is also within the initial radio range of j. An acyclic subgraph $T = (V, E')$ of G is defined to restrict the node radio ranges. An edge $(i, j) \in E'$ is incident on nodes i and j. The radio range of node i is r_i ($\leq r_{max}$), which will be set to the same length as the longest edge of all the incidents on node i. N_i is a set of neighboring nodes of node i, whose elements are within r_i. $F_i (\subseteq N_i)$ is a set of family nodes, whose members are either a parent or children of node i or both. A cost (weight) function on an edge (i, j), c_{ij} is the sum of all the transmitting, receiving, overhearing, and idle energy consumed by all the related nodes when a message is sent from node i to node j.

6.2.2 Energy Model

In wireless sensor networks, communication energy is consumed when a message is sent from one node to another. In general, energy is consumed when a node is sending, receiving, and overhearing a packet or a node is idle. Our energy model is based on [38], which considers energy consumption in overhearing messages. Overhearing energy is consumed when a node receives a message that is not destined to it due to being within the radio range of a sender. Energy consumption in one bit data transmission ($\mathcal{E}_{tx}$) is proportional to distance raised to the power of 2 to 4 due to path loss depending on the medium.

$$\mathcal{E}_{tx} = \mathcal{E}_{txelec} + \epsilon d^{\alpha} \tag{6.1}$$

where, $\mathcal{E}_{txelec}$ is the energy consumption by transmitter electronics, ϵ is an amplifier characteristic constant, d is a distance between two communicating nodes when the radio range of a sender is assumed to be the same as the distance, and α is the path loss whose value is 2 to 4. In this work, we assume that the value of α is 2. Energy consumption in reception ($\mathcal{E}_{rx}$) is independent of the distance and it needs only $\mathcal{E}_{rxelec}$, which is consumed by receiver electronics.

$$\mathcal{E}_{rx} = \mathcal{E}_{rxelec} \tag{6.2}$$

For energy consumption in overhearing ($\mathcal{E}_{oh}$), an overhearing node consumes the same amount of energy as when a node receives one bit data.

$$\mathcal{E}_{oh} = \mathcal{E}_{rxelec} \tag{6.3}$$

We let $\mathcal{E}_{txelec} = \mathcal{E}_{rxelec}$ as assumed in [37] and use the symbol $\mathcal{E}_{elec}$ for both. We assume that a collision avoidance MAC protocol such as IEEE 802.11 or S-MAC is used *. In these protocols, in order to send a message from one sensor node to the other, RTS (ready to send), CTS (clear to send), DATA (actual message to be transmitted), and ACK (acknowledgement) packets are necessary to avoid DATA packet collision. Because of this reason, two communicating nodes have to be symmetric in terms of reachability. That means that each one of the two nodes should be able to send a message to the other node. We give an example when a message is transmitted from node i to node j through a multi-hop path in figure 6.1. In figure 6.1(a), after node i sends a RTS packet to node j, since node i is out of bound of the radio range of node j, a CTS packet that is in response to the RTS packet cannot reach node i. Therefore, we have to assign radio range on nodes so that both nodes on each link can communicate with each other. Hence, node i sets its radio range so that it can reach not only its children to send

*Other protocols are also used in sensor networks, but our work focuses on this protocol

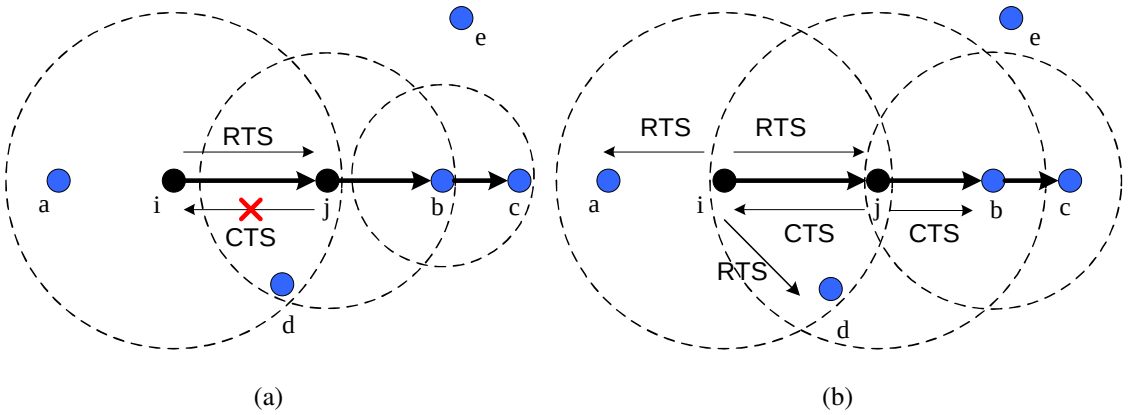


Figure 6.1: Two types of radio range for RTS and CTS (a) asymmetric (b) symmetric

CTS and ACK packets, but also its parent to send RTS and DATA packets. Therefore, its radio range must cover the length of the longest edge that connect node i to nodes in F_i .

To calculate the amount of energy that is associated with an edge, we need to add total energy consumption related to completing message transmission from one node to the other. In figure 6.1(b), the cost on an edge (i, j) is the sum of transmitting energy of RTS and DATA and receiving energy of CTS and ACK at node i, transmitting energy of CTS and ACK and receiving energy of RTS and DATA at node j, receiving energy of RTS at node a, receiving energy of CTS at node b, and overhearing energy of RTS at node d. In this case, the overhearing energy of CTS (from node j) at node d is excluded since node d goes to sleep mode after receiving RTS from node i due to NAV (Network Allocation Vector) that indicates a period for sleeping. Also, we need to consider idle energy of node c and e, which is not directly related to communication between node i and j.

In order to consider energy for idle time, we assume our simplified duty cycle that looks like figure 6.2. Since our energy model is based on energy consumption (Joule) per one bit transmission, we convert idle time into bits, whose amount of data is assumed to be transferred for the given idle time. We assume one cycle is corresponding to time for which 300 bytes (2400 bits) data is transferred. Therefore, if we assume DATA is 200 bytes, RTS, CTS, and

ACK are 10 bytes, and if we assume that converted bytes for one duty cycle (b_c) is 300 bytes, then converted bytes for idle time is 70 bytes (560 bits). We set b_w as the sum of bytes for RTS, CTS, DATA, and ACK. We set k as the ratio of idle energy to receive energy. Typically, this ratio varies from 0.5 to 1 depending on wireless devices and experiments [87, 88, 89]. This is summarized in table 6.1 with the actual energy consumption in each case.

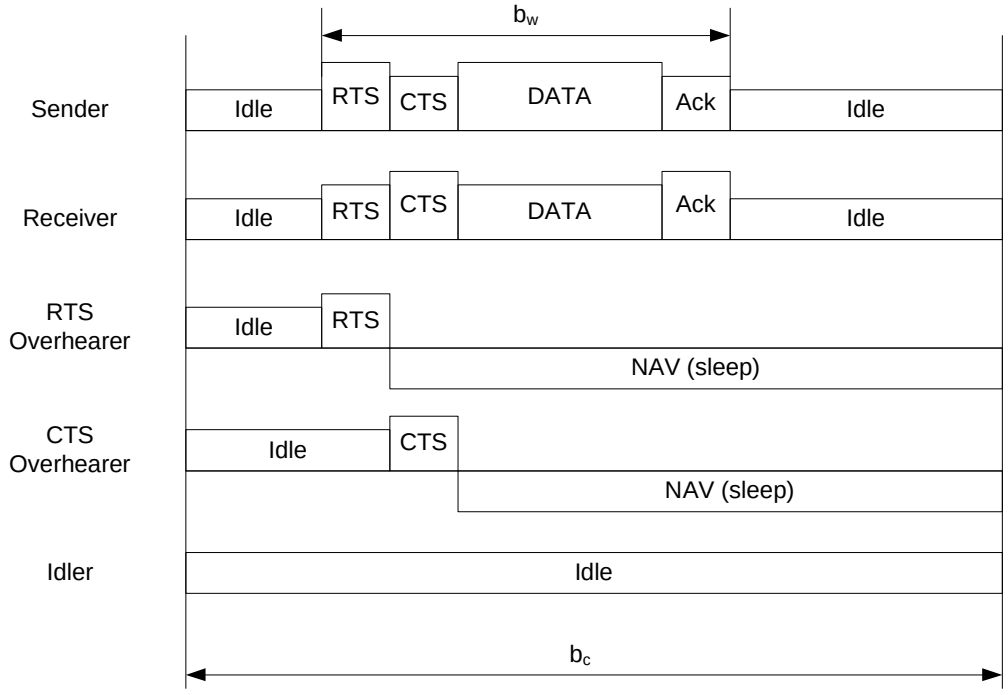


Figure 6.2: Cycle diagram for different nodes with energy consumption

Generally, the total amount of energy associated with the edge (i, j), $\mathcal{E}_{ij}$ is represented as follows:

Table 6.1: Energy consumption in each case. Initially $\mathcal{E}_{idle}$ for node b is $k(\frac{b_c-b_w}{2}+|RTS|)\mathcal{E}_{elec}$. But it is changed for simplification.

node	$\mathcal{E}_{tx}$ (energy for transmission)	$\mathcal{E}_{rx}$ (energy for reception)	$\mathcal{E}_{idle}$ (energy for idle)
i	$\|RTS+DATA\|(\epsilon {r_i}^2+\mathcal{E}_{elec})$	$\|CTS+ACK\|\mathcal{E}_{elec}$	$k(b_c-b_w)\mathcal{E}_{elec}$
j	$\|CTS+ACK\|(\epsilon {r_j}^2+\mathcal{E}_{elec})$	$\|RTS+DATA\|\mathcal{E}_{elec}$	$k(b_c-b_w)\mathcal{E}_{elec}$
a,d	N/A	$\|RTS\|\mathcal{E}_{elec}$	$k\frac{b_c-b_w}{2}\mathcal{E}_{elec}$
b	N/A	$\|CTS\|\mathcal{E}_{elec}$	$k\frac{b_c-b_w}{2}\mathcal{E}_{elec}$
c,e	N/A	N/A	$kb_c\mathcal{E}_{elec}$

$$\begin{aligned}\mathcal{E}_{ij} &= |RTS+DATA|(\epsilon {r_i}^2+\mathcal{E}_{elec})+|CTS+ACK|(\epsilon {r_j}^2+\mathcal{E}_{elec})\\ &+ |RTS+CTS+DATA+ACK|\mathcal{E}_{elec}\\ &+ n_1|RTS|\mathcal{E}_{elec}+n_2|CTS|\mathcal{E}_{elec}\\ &+ 2k(b_c-b_w)\mathcal{E}_{elec}+(n_1+n_2)k(\frac{b_c-b_w}{2})\mathcal{E}_{elec}\\ &+ (N-n_1-n_2-2)kb_c\mathcal{E}_{elec} \qquad (6.4)\end{aligned}$$

where, r_i and r_j are the radio ranges of nodes i and j respectively, n_1 is the number of neighbors that overhear the RTS packet, n_2 is the number of neighbors that overhear the CTS packet, N is the total number of nodes in the sensor field, and |packet| is the size of the 'packet' in bits. If we assume that $|RTS|=|CTS|$, and we simplify the equation (6.4), we get following equation

$$\mathcal{E}_{ij} = \alpha\epsilon {r_i}^2+\beta\epsilon {r_j}^2+\gamma n\mathcal{E}_{elec}+\delta\mathcal{E}_{elec} \qquad (6.5)$$

where, n is the number of nodes that receive RTS or CTS packet (n_1+n_2), α is $|RTS|+|DATA|$, β is $|CTS|+|ACK|$, γ is $|RTS|-k/2(b_c+b_w)$, and δ is $2(|RTS|+|CTS|+|DATA|+|ACK|)+k(Nb_c-2b_w)$.

Finally, we add up the energy consumption by all the edges in a tree T to get the energy consumption by the whole network, $\mathcal{E}$.

$$\mathcal{E} = \sum_{(i,j) \in E'} \mathcal{E}_{ij} \tag{6.6}$$

6.3 Problem Definition

Given sensor nodes with maximum radio range and a designated sink node, we would like to gather data from all the sensor nodes and relay the data to the sink node using in-network aggregation so that the sink node gets one result data. An example query in this case is to report average temperature of all the sensors every 10 minutes for 1 month. When a temperature value is passed up to its parent, the parent aggregates the data from all of its children by counting the number of its children and by summing up their temperatures. Then, it adds its own temperature and augments the count by one, and passes this information to its parent. By repeating this, the sink node can finally know the total number of nodes in a tree and process the total sum of the temperature values in the tree. It can then calculate the average temperature. This in-network aggregate processing is proposed and well described in [28]. In-network processing guarantees that all the sensor nodes can send a packet only once to its parent for one cycle of a query. Therefore, the number of transmissions of actual data packets is the same as the number of edges in the EDG-tree. Our problem is to find a tree which minimizes the sum of energy cost on all its edges.

Although most researchers provide equations to estimate energy consumption at a node, our model estimates energy consumption on a link between two nodes. The main reason is that our heuristic algorithms for incrementally constructing an energy-efficient tree require edge-based energy measurement. Our model will produce the same accuracy as a node-based model once a tree structure is determined, but we use more accurate formulas to estimate energy consumption during the process of constructing an energy-efficient tree structure. This will be explained in more detail in section 6.4. Now, we provide a decision form of our EDG-tree problem to prove that the problem is NP-complete.

Energy-efficient Data Gathering Tree (EDG-tree)

Instance: A graph $G = (V, E)$, a subgraph of G, a tree $T = (V, E')$, whose $|E'| = |V| - 1$ and having no cycle, where $E' \subset E$, an edge $(i, j) \in E'$, which is incident on node $i(\in V)$ and node $j(\in V)$, a radio range assignment on a node i, $r_i : V(T) \rightarrow R^+$, the number of overhearing nodes from i and j, n_{ij}, a cost (weight) function on an edge (i, j), $c_{ij}(r_i, r_j, n_{ij}) : E'(T) \rightarrow R^+$, which is defined in equation (6.5), and a positive constant $B \in R^+$.

Question: Is there a cost assignment set, $C = \{c_{ij} | \forall (i, j) \in E'\}$ such that a tree T has $c_{ij}(r_i, r_j, n_{ij})$ for all $(i, j) \in E'$, whose r_i (or r_j) is the same length as the longest one of all the edges of node i (or j) and, which satisfies $\sum_{(i,j)\in E'} c_{ij}(r_i, r_j, n_{ij}) \leq B$?

T represents a general tree, which can be any subgraph of G, constrained to be a structure that satisfies two conditions, i) $|E'| = |V| - 1$, ii) having no cycle. Our goal is to find an energy-efficient T that satisfies the additional constraints specified in the Question above.

To prove the NP-completeness of the EDG-tree problem, we show the problem is equivalent to an existing NP-complete problem, Min-Power Symmetric Connectivity (MSC), which is shown as NP-hard in [64]. We provide a proof by restriction [68] that shows that the EDG-tree problem contains the MSC problem as a special case. That means, if we add restrictions on the instances of the EDG-tree problem, then the restricted problem can be equivalent to the MSC.

Min-power Symmetric Connectivity (MSC)

Instance: An edge weighted graph $G = (V, E, c)$, undirected edge between nodes u and v, uv, the cost $c(uv)$ of an edge $uv \in E$ corresponding to the (symmetric) power requirement $p(u, v) = p(v, u)$, maximum cost edge incident to u in a spanning tree T for a node $u \in V$, uu_T, i.e., $uu_T \in T$ and $c(uu_T) \geq c(uv)$ for all $uv \in T$, a range assignment for V, $r : V \rightarrow R^+$,

the cost of r, $c(r) = c(uu_T) = |uu_T|^2$, where $|uu_T|$ denotes the Euclidean distance between u and u_T, and a positive constant $B \in R^+$.

Question: Given a connected edge-weighted graph $G = (V, E, c)$, is there a spanning tree T of G, which satisfies the power cost of the spanning tree T, $p(T) = \sum_{u \in V} c(uu_T) \leq B$?

Proof. First of all, we show that the instances of the EDG problem is sufficient to include those of the MSC problem. Common instances of both problems include a tree, an edge, and a radio range assignment. Implicit power requirement in EDG includes that in MSC since EDG assigns power to nodes so that it satisfies the condition, $r_i \geq |ij|$ and $r_j \geq |ij|$. The number of overhearing nodes is only included in instances of EDG but not in MSC, and is used in the cost function. The differences between the problems are in the definition of the cost (weight) functions on edges. The first difference between the two cost functions is that the cost function has three arguments in EDG and one argument in MSC. The second difference is that the EDG problem minimizes the sum of costs assigned on edges, while the MSC problem minimizes the sum of costs assigned on nodes.

Based on these instances, now we show that the EDG problem contains the MSC problem as a special case. In MSC, the purpose is to find a tree that minimizes the sum of $c(uu_T)$ of all the *nodes* in the network, which means that the cost is assigned on the nodes. In this case, the cost of one node i is $|ii_T|^2$, i.e., $|ai|^2$ as shown in figure 6.3(a). Meanwhile, in EDG, the purpose is to find a tree that minimizes the sum of $c(r_i, r_j, n_{ij})$ of all the *edges* in the network, which means energy is consumed by all the edges. This sum of the cost of *edges* can be represented as the energy consumed by *nodes*. When we consider the case where node i sends data to node j, the cost on the edge (i, j) is $\alpha\epsilon {r_i}^2 + \beta\epsilon {r_j}^2 + \gamma n\mathcal{E}_{elec} + \delta\mathcal{E}_{elec}$ based on equation (6.5). Since this cost is consumed by an edge, we need to separate it into two nodes so that cost can be assigned on a node. The cost on node i is the sum of energy that is necessary for sending RTS and DATA packets to node j, $(|RTS| + |DATA|)(\mathcal{E}_{elec} + \epsilon {r_i}^2)$, for receiving CTS and

ACK packets from node j, $(|CTS|+|ACK|)\mathcal{E}_{elec}$, and for idle time, $k(b_c-b_w)\mathcal{E}_{elec}$. Similarly , when node i receives data from another node, say a, the cost on node i is the sum of energy that is necessary for receiving RTS and DATA packets from node a, $(|RTS|+|DATA|)\mathcal{E}_{elec}$, for sending CTS and ACK packets to node a, $(|CTS|+|ACK|)(\mathcal{E}_{elec}+\epsilon {r_i}^2)$, and for idle time, $k(b_c-b_w)\mathcal{E}_{elec}$. Also, node i can overhear neighboring nodes communicate, such as data from j to d. The cost for overhearing is $(m_2+m_3)(|RTS|+k\frac{b_c-b_w}{2})\mathcal{E}_{elec}$, where m_2 and m_3 are the number of RTS and CTS sending nodes that node i overhears. Finally, $m_4 k b_c \mathcal{E}_{elec}$ is added to consider a situation that node i does not do anything but idle listening, where m_4 is the number that node i works as just a idle listener. This is shown in figure 6.3(b), where r_i is represented as $|ai|$, which is the actual value for r_i.

Generally, a node i sends data just once to its parent node in the tree and receives data multiple times depending on the number of its child nodes. In the equation 6.7, we set m_1 as the number of children that node i has. Therefore, the cost for node i can be rewritten as

$$\begin{aligned} c(i) &= (|RTS+DATA|)(\mathcal{E}_{elec}+\epsilon {r_i}^2) \\ &+ (|CTS+ACK|)\mathcal{E}_{elec}+k(b_c-b_w)\mathcal{E}_{elec} \\ &+ m_1\{(|CTS+ACK|)(\mathcal{E}_{elec}+\epsilon {r_i}^2) \\ &+ (|RTS+DATA|)\mathcal{E}_{elec}+k(b_c-b_w)\mathcal{E}_{elec}\} \\ &+ (m_2+m_3)\{|RTS|\mathcal{E}_{elec}+k/2(b_c-b_w)\mathcal{E}_{elec}\} \\ &+ m_4 k b_c \mathcal{E}_{elec} \end{aligned} \tag{6.7}$$

If we still assume that $|RTS|=|CTS|$ and rewrite equation (6.7) in terms of ${r_i}^2$,

$$c(i)=\alpha {r_i}^2+\beta \tag{6.8}$$

where, $\alpha=\epsilon(|RTS+DATA+CTS+ACK|)$, and $\beta=\mathcal{E}_{elec}\{(m_1+1)|RTS+CTS+DATA+ACK|+(m_1+1/2(m_2+m_3)+1)k(b_c-b_w)+(m_2+(k+1)m_3)|RTS|+m_4 k b_c\}$.

This means that the cost equation (6.8) is a general form of cost function of MSC, when α is 1 and β is 0, which reduces the cost $c(i)$ of the EDG to exactly the same as $c(i)$ of MSC. So far, we have shown that the cost function of EDG on an edge that has three arguments can be rewritten as that of MSC on a node. Therefore, we show that the MSC problem is a special case of the EDG problem. This concludes the proof. □

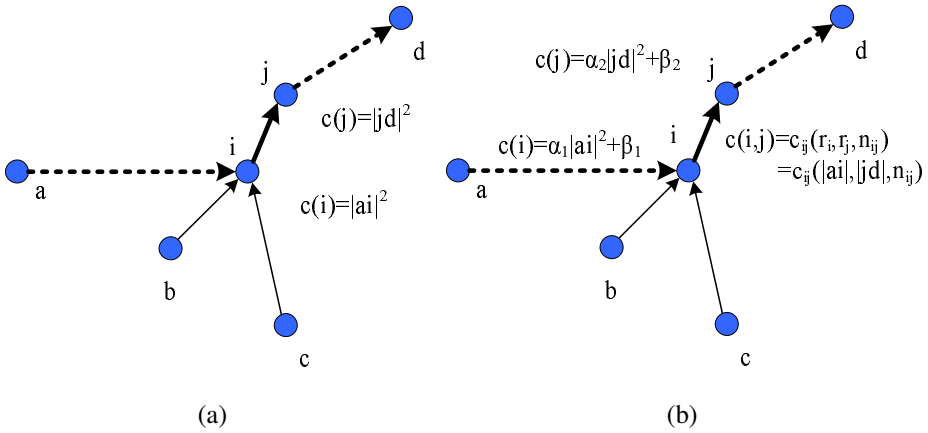


Figure 6.3: Equivalent cost separation from edges to nodes. Dotted line represents the longest edge of node i and j respectively. (a) Cost of MSC (b) Cost of EDG

6.4 Heuristic Algorithms

In this section, we propose two heuristic algorithms that try to minimize the sum of edge costs. One of the two algorithms that we propose backtracks one step (BT1) and the other backtracks two steps (BT2). BT1 and BT2 use backtracking when the algorithms span the nodes step by step to create a tree as an augmentation to Prim's minimum spanning tree (MST) algorithm. The backtracking implies that the algorithm can delete an edge that was already chosen for a local optimal solution at a previous step and replace it with a new edge since the new edge can minimize the total cost of a tree at the current step. These are later compared to existing algorithms that were proposed in [41] in section 6.5.

For cost, we choose an edge based cost instead of node based cost. Once the edge (i, j) is determined as part of a spanning tree (see figure 6.1(b)), we can easily calculate energy consumption on the edge by nodes i and j, when data is transferred from node i to node j. This includes energy for transmitting RTS and DATA and receiving CTS and ACK at node i and for transmitting CTS and ACK and receiving RTS and DATA at node j. The overhearing energy consumed by neighbors of either node i or j can be easily added to that edge (see table 6.1). This is possible because nodes i and j will know their radio ranges during the spanning tree construction. However, if we use node based cost, we have to consider all kinds of energy that one node consumes. But nodes that are not in the spanning tree do not know yet their radio range since their family nodes (F_i) are not determined, and thus overhearing energy consumed by some nodes in the spanning tree cannot be calculated exactly. That means that a node in the spanning tree cannot determine how many nodes that are not yet included in the spanning tree will reach it. This makes the edge based cost more appropriate than the node based one, since it does not need to know as much information about neighboring nodes. The cost function has three arguments as defined in equation (6.5), which are radio range of a sender (r_i), radio range of a receiver (r_j), and the number of neighboring nodes that overhear any RTS or CTS packet (N).

6.4.1 Backtracking-1 Algorithm

We now propose a heuristic algorithms for the EDG problem since the problem is proven to be NP-complete. The BT1 (BackTracking-1) is based on Prim's MST in that it creates a spanning tree by adding incrementally a node whose edge cost to the other nodes in a spanning tree is minimized. The backtracking feature is added to Prim's algorithm to support and correct cases that the basic MST algorithm does not consider. This is necessary because our cost model refers to both sender's and receiver's radio ranges to use an edge based cost. Due to this reason, even though a node is included in a current spanning tree, once the node's radio range needs to

be increased † to span another node, we have to recalculate the cost of all the edges attached to this node. This is because when the radio range is increased, one of the arguments in the cost function is changed, which affects the cost. The BT1 considers this situation and gives a chance to backtrack to its previous step to find out if there is any edge that is more energy-efficient. If there is, it changes the edges so that they can be the current optimal solution while they are creating the spanning tree. Otherwise, it keeps the current spanning tree. This situation is illustrated in figure 6.4.

Given a partial tree $T = (V, E')$ at step n-1 (figure 6.4(a)), the edge set of the current spanning tree, E' is $\{(a, b), (b, c), (b, d)\}$ (edges in a current spanning tree are represented by thick lines). The numbers on the edges represent the distance between two nodes and numbers in parentheses are the cost on the edges. Assume that node a is a root. Now, node d spans node e, because the edge (d, e) is the least cost among the candidate edges, $(d, g), (d, e), (b, e)$, and (c, e). Next, the edge (e, f) is selected and added to E' at step n-1 (figure 6.4(b)). In this case, since node f is set to be a child of e, the radio range of node e is increased from 47 to 71. That requires that the cost of edge (d, e) should be recalculated because the edge cost considers radio range of both nodes d and e. Even though the cost for (d, e) is increased to 280, it still remains less than any other candidate edges to be included in set E', $(b, e), (c, e), (d, g)$ and (e, g). At the next step (figure 6.4(c)), node e tries to span g since it is less than the cost of (d, g). When node g is included in the set E', the radio range of node e is again increased from 71 to 90, thus changing the cost on (d, e) again (280 to 340). The total cost of the spanning tree can be decreased when the edge (b, e) is chosen instead of (d, e). The total cost of the spanning tree in the figure 6.4(c) is 1809, while that in the figure 6.4(d) is 1808. BT1 chooses the tree in figure 6.4(d) since its cost is less than that in figure 6.4(c). In order to consider this kind of case, we propose an algorithm that can backtrack one step and rebuild part of a tree.

†But it is still less than the node's maximum radio range

Figure 6.5 shows the pseudocode for the BT1 algorithm. Backtracking begins at line 16 when the radio range of a node in a spanning tree is increased because of a newly added node. For backtracking one step, it is necessary to have at most E iterations for the outer loop. When a cycle is created, we have to find a case that consumes the least energy by deleting one edge from the cycle. If we let E_C be the set of edges forming a cycle and let V_C be the set of nodes on the cycle, it takes $|E_C V_C|$ time to test all the cases. Therefore, the backtracking feature cost is $O(VE^2)$ in the worst case, since $|E_C| \leq |E|$ and $|V_C| \leq |V|$.

6.4.2 Backtracking-2 Algorithm

Next, we propose a second algorithm BT2 (BackTracking-2), which is a variation of BT1. BT2 backtracks up to two previous steps to try to find better solutions. Let (u_{min}, v_{min}) be an edge whose cost is minimum among all other candidate edges. Node u_{min} is in a current spanning tree and node v_{min} is not. Let node v_1 be either u_{min} or v_{min}. Let u_2 be one of nodes that is in the current spanning tree and is within a maximum radio range of v_1, but is different from u_{min}. When v_1 looks back at u_2, BT2 connects them as in BT1. In BT2, if the radio range of u_2 is also increased due to the newly created edge, (v_1, u_2), then the node u_2 backtracks again and tries to find u_3, which is a node in the spanning tree and within maximum radio range of u_2 and so it can be connected to u_2. BT2 tries to find the minimum cost spanning tree by comparing all the possible sub-trees created by combination of all the (v_1, u_2) and (u_2, u_3) and deletion of appropriate edges to avoid cycles. BT2 always maintains the graph so that it can meet the requirement of being a spanning tree, and the algorithm calculates the total cost of the spanning tree. In other words, while the edges (v_1, u_2) and (u_2, u_3) are being created, BT2 deletes edges from a cycle one by one so that it can find the minimum cost spanning tree. Basically, since BT2 repeats backtracking one more step, the running time for backtracking-2 is the square of that of BT1. In the worst case, the backtracking-2 feature takes $O(V^2E^4)$, which is still a polynomial time.

6.5 Experimental Results

In this section, we show various results of our simulation. As we mentioned in the section 6.2.2, there are many parameters that can affect the amount of energy consumed by all the nodes in the network. We simulated Prim's MST algorithm, Energy Conserving Routing Tree (ECRT) and Local Optimization (LOCAL-OPT) as proposed in [41], in order to compare them with BT1 and BT2. Also, we implemented heuristic algorithms such as genetic algorithm (GA) and simulated annealing (SA) as described in sections 3.2 and 3.3 for the purpose of comparison. The parameters that we varied are the number of nodes in a network, size of data packet, the amplifier constant, the electronic energy, and the ratio between idle energy and receiving energy. Our simulation was done in JAVA and executed on an Intel Pentium 4 (1.6 GHz) computer. In each experiment, we gave 10 different seeds and got an average total energy and running time.

Prim's MST creates a spanning tree by incrementally adding a node whose edge cost is minimum to the current tree. In our simulation, Euclidean distance between two nodes is used as the cost of Prim's MST. Since MST is proven to be 2-approximation algorithm for the MSC problem in [40], we basically compare other algorithms with MST. ECRT and LOCAL-OPT ([41]) are proposed to maximize the lifetime of the resulting tree. In our simulation, these are slightly modified to minimize energy consumption, which is the goal of this paper. At every step, ECRT determines the energy consumption of the tree constructed so far, and adds an edge that will minimize the energy consumed by the resulting new tree instead of considering the lifetime. LOCAL-OPT switches the parent of the current node when the new tree with the new parent consumes less energy than before. This also considers minimizing energy consumption instead of maximizing the lifetime.

6.5.1 Overview of simulation

We compared our two algorithms, BT1 and BT2, with three algorithms: MST, ECRT, LOCAL-OPT, GA, and SA. We varied the number of nodes from 20 to 100 when data size is 10 bytes and 200 bytes, electronic energy ($\mathcal{E}_{elec}$) is 50.0 nJ/bit, amplifier constant (ϵ) is 0.01 $nJ/bit/m^2$, and the ratio between idle energy and receiving energy is 0.5. Figure 6.6 shows the energy consumption by all the nodes in the network when the nodes send their data once to their parents in response to a data gathering query. ECRT can be regarded as similar to BT1 or BT2 without a backtracking feature. As we can see in figure 6.6, LOCAL-OPT consumes more energy than the other six algorithms regardless of the number of nodes. In addition to it, since the total energy consumed by these six algorithms (Prim's MST, ECRT, BT1, BT2, GA, and SA) are very close to each other, we compare only these six algorithms to take a closer look at their results.

6.5.2 Percentage Gain

Percentage gains are measured by comparing the energy consumption of each algorithm with Prim's MST algorithm using the equation: $(MST - cost)/MST \times 100$, where $cost$ is the energy consumption by BT1, BT2, ECRT, GA, or SA. That gives the percentage gains compared to Prim's MST. In the figure 6.7(a), when the data size is 10 bytes, BT2 outperforms BT1 in all the cases of different numbers of nodes, and BT1 outperforms ECRT in most of the cases. Also, BT2 outperforms all the other algorithms except the number of nodes are 20 and 40. This is because the number of nodes are too small to reflect the performance of BT2 algorithm when the number of nodes are 20. Also, GA and SA are not optimized to solve this EDG problem, they show irregular percentage gain depending on different number of nodes. This makes GA outperform the other algorithms when the number of nodes are 40.

To examine the effect of varying data size further, we evaluated percentage gain for different data sizes, from 10 to 200 bytes. As we can see in figure 6.7(b), the percentage gain

of all five algorithms are decreasing as the data size increases except GA. This means that the more idle time and the less energy for data communication, the more percentage gain for all the algorithms. BT1 is almost same as ECRT, but outperforms GA or SA in most cases. BT2 outperforms all the other algorithms regardless of data size. This test is done when the number of nodes is 80.

Next, we gave various values of $\mathcal{E}_{elec}$, electronic energy, which can vary based on the type of sensor (see figure figure 6.8(a)). Initially, we used 50 nJ/bit from [37], but we also gave different values such as 0.5, 1, 5, 10, 20, and 30 nJ/bit. This is also done when the number of nodes is 80. BT2 outperforms all the other algorithms regardless of $\mathcal{E}_{elec}$. BT1 is also better than ECRT in all the different electronic energy values. But, behavior of SA, which is not optimized for solving EDG problem, BT1 is not better than SA when electronic energy is 0.5, 1, and 5 nJ/bit.

We also tested percentage gain with respect to different amplifier constant (ϵ) from 0.01 to 0.1 $nJ/bit/m^2$ (see figure 6.8(b)). [37] used 0.1 as the value of ϵ. BT2 algorithm outperforms all the other algorithms when the value is low (0.01-0.05). But as the amplifier constant value increases, percentage gain of BT2 decreases and GA or SA have better percentage gain than BT2 when the value is high (0.06-1.0).

Finally, we tested the effect of idle energy (see figure 6.9). In order to cover possible ratio of idle energy to receiving energy, we consider various ratios that includes existing experimental values [87, 88, 89]. Typically, idle energy is less than receiving energy, which can be represented as a ratio less than 1.0. Therefore in most reasonable ratios, which is less than 1.0, BT2 algorithm outperforms all the other algorithms. Also, BT1 outperforms the other three algorithms. We showed the ratio that is bigger than 1.0 to give information when idle energy passes receiving energy.

6.5.3 Running time

Even though we propose a heuristic algorithm, it is necessary to measure actual running time for average case analysis. Also we need to consider the trade-off between cost and running time. We have seen so far that BT2 creates a tree that uses the least energy cost with BT1 and MST in all cases of the different parameters, number of nodes, data size, electronic energy, and amplifier constant. BT2 saves more than BT1, and BT1 saves more than MST. The running time for these algorithms is a reverse order of cost with respect to the number of nodes (figure 6.10). This shows the trade-off between cost and running time in our simulation, since the most cost saving algorithm, BT2, takes the most time among the algorithms. However, since the algorithm is only run once, and data gathering will be carried out many times, the savings in energy cost is more important than the running time of the algorithms that create the data gathering tree.

6.6 Summary

In this chapter, we defined the EDG-tree problem as a minimum cost spanning tree problem with edge based cost and we proved that the problem is NP-complete. Unlike existing data gathering tree, we considered two-way data communication including overhearing and idle energy as we calculate costs on an edge in order to perform a more realistic simulation. Many parameters that affect real energy consumption were also considered to have more precise simulation results. As heuristic algorithms for the EDG problem, we proposed BT1 and BT2 algorithms that backtrack one or two steps so that they can find better local solution while they are creating a spanning tree. BT1 and BT2 outperform existing algorithms in terms of cost saving.

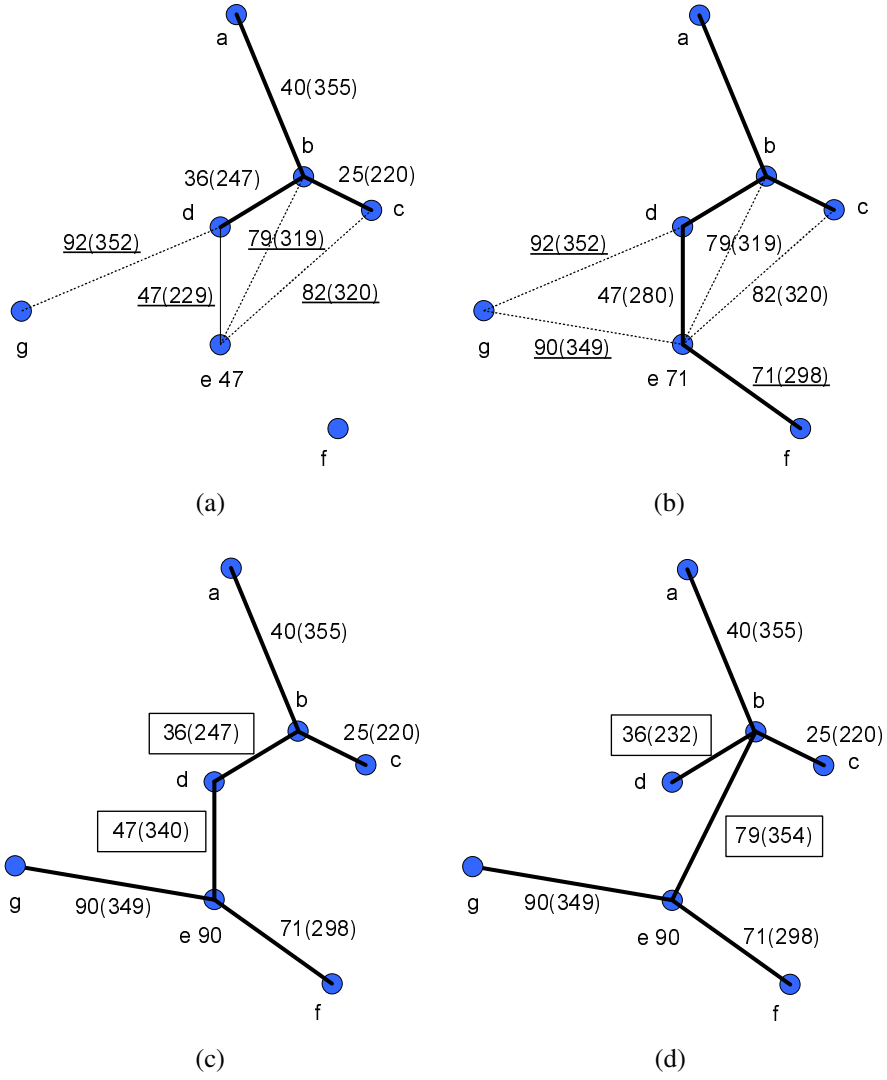


Figure 6.4: Backtracking model (thick line: edges in a spanning tree, thin line: the least cost edge among the candidate edges, dotted line: candidate edges) (a) step n-1 (i) (b) step n-1 (ii) (c) step n (i) (d) step n (ii)

Algorithm 2: Backtracking-1 (BT1)

Input: G=(V,E)
Output: T=(V_T,E_T)

```
 1: V_T ← {r}, E_T ← Ø, step ← 0 // r; root node
 2: while |V_T| ≠ |V| do
 3:    step ← step + 1
 4:    w_min ← infinity, w_uv ← 0
 5:    for all u ∈ V_T do
 6:       r_u ← max {dist(u,u_T)} for all (u,u_T) ∈ E_T , where u_T ∈ V_T
 7:       for all v ∈ adj(u) in V, but not in V_T do
 8:          r_v ← dist(u,v)
 9:          r_u ← max {r_u,r_v}
10:          w_uv ← weight (r_v,r_u,#overhear)
11:          if w_uv < w_min then // find u and v whose w_uv is minimum
12:             (u_min,v_min) ← (u,v)
13:          end if
14:       end for
15:    end for
16:    if dist(u_min,v_min) > r_u then // backtracking begins
17:       v_1 ← u_min or v_min , E_S ← E_T ∪ {(u_min,v_min)}
18:       for all (v_1,u_2) ∈ E and (v_1,u_2) ∉ E_T , where u_2 ∈ V_T do
19:          E_S ← E_S ∪ {(v_1,u_2)}, V_S ← V_T ∪ {v}
20:          find a set of edges and nodes, E_C and V_C,which make a cycle including (v_1,u_2)
21:          for all (i,j) ∈ E_C do
22:             E_C ← E_C \ {(i,j)}, E_S ← E_S \ {(i,j)}
23:             for all n ∈ V_C do // radio range adjustment
24:                r_n ← max { dist(n,n_T) } for all (n,n_T) ∈ E_S , where n_T ∈ V_T
25:                calculate cost for E_S
26:                E_min ← E_S, whose cost is minimum
27:             end for
28:          end for
29:       end for
30:    end if
31:    V_T ←V_T ∪ {v_min}, E_T ← E_min
32: end while
33: return T(V_T,E_T)
```

Figure 6.5: The pseudocode of BT1 algorithm

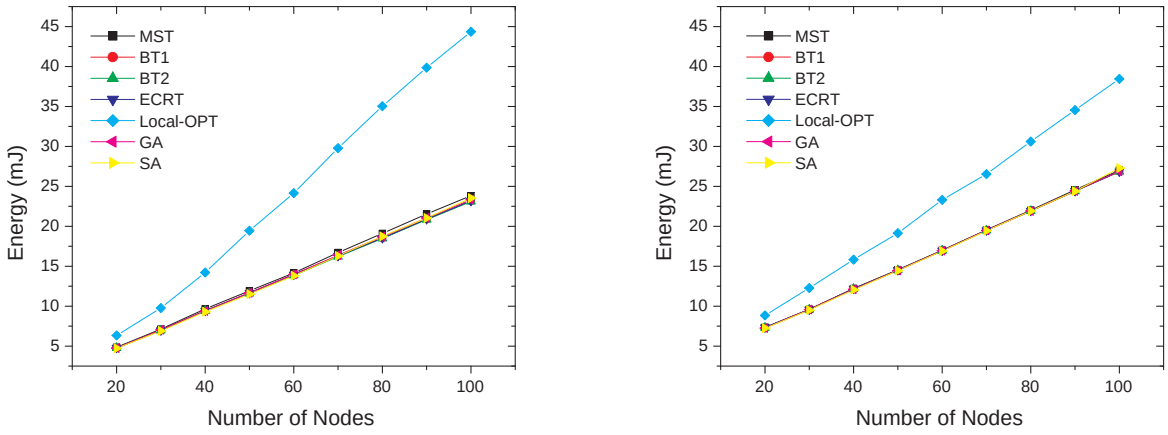


Figure 6.6: Overview of energy consumption (a) energy (data size = 10 bytes) (b) energy (data size = 200 bytes)

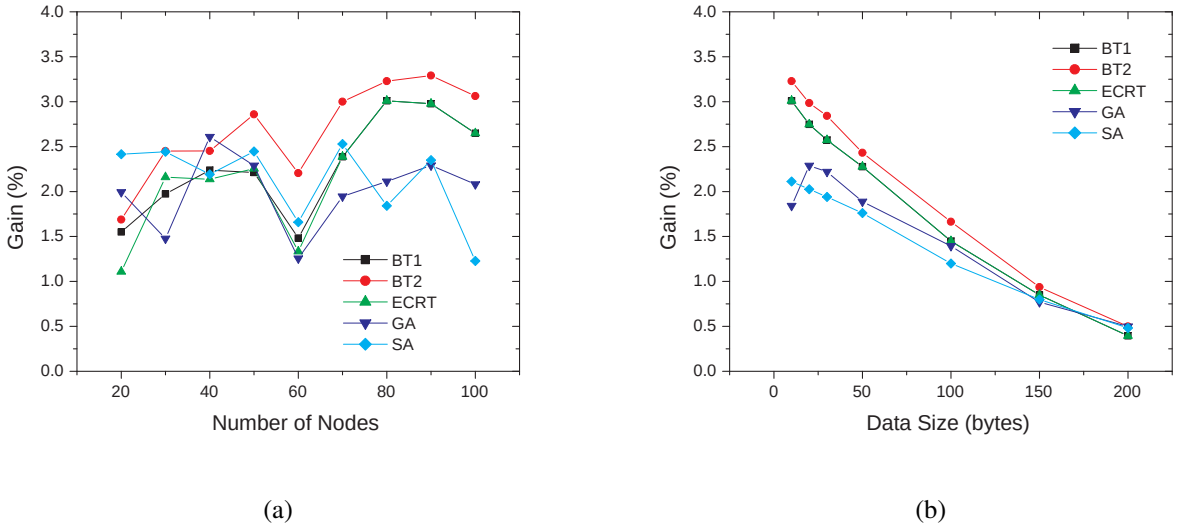
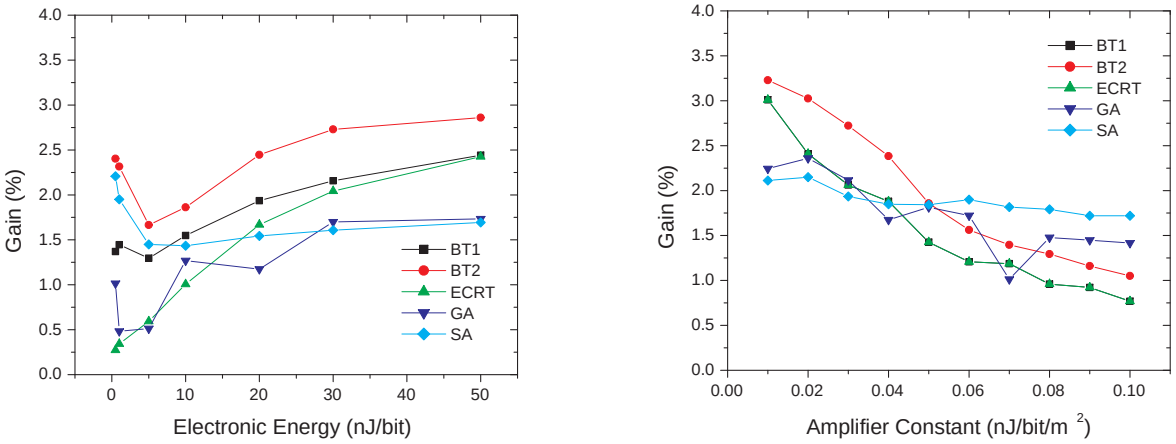


Figure 6.7: Percentage gain for various number of nodes and data size (a) percentage gain in terms of number of nodes (b) percentage gain in terms of data size



(a) (b)

Figure 6.8: Percentage gain for various electronic energy and amplifier constant (a) percentage gain in terms of electronic energy (b) percentage gain in terms of amplifier constant

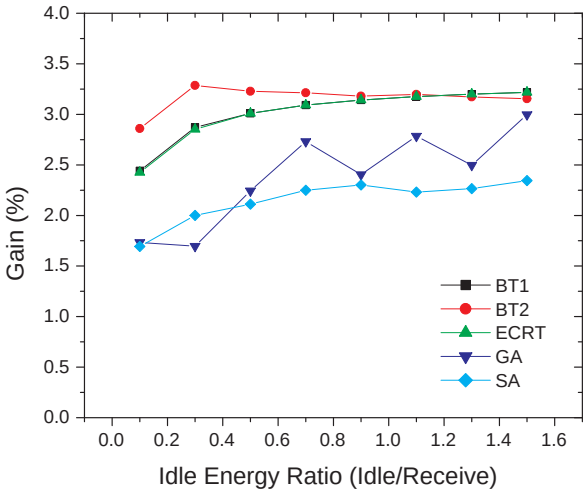


Figure 6.9: Percentage gain for various ratio of idle energy to receiving energy

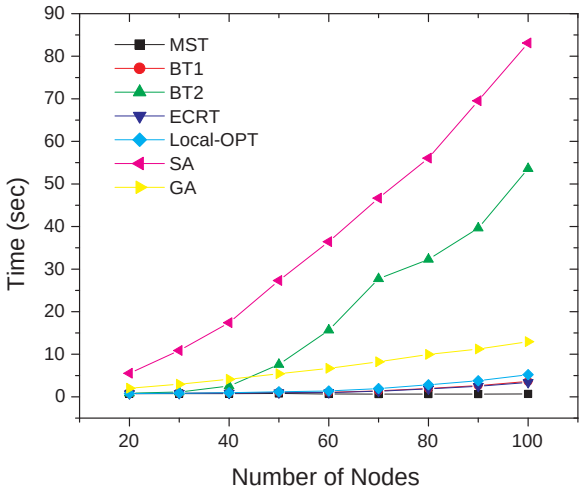


Figure 6.10: Running time

CHAPTER 7

CONCLUSIONS AND FUTURE RESEARCH

Throughout this thesis, first we studied characteristics of queries in wireless sensor networks and various network schemes that include storage schemes and routing protocols. We studied what kind of queries are suitable for what kind of network schemes when we consider the number of transmissions, energy consumption, end-to-end delay, life span, and local storage capacity. Second, we proposed sectioned tree that saves energy when we disseminate spatial query to an area of interest. The proposed scheme, sectioned tree, also saves energy when the network gathers data and report them to a base station. It uses existing method for local tree construction, which is not guaranteed to save energy. Third, because of this reason, we proposed heuristic algorithms that backtrack one or two steps to build an energy-efficient local tree to gather data from the networks. In this case, we considered two-way communication, overhearing energy, and idle energy, so that the algorithm and simulation can reflect more precise real world situation. For the second and third work, we focused on the spatial and temporal query that report sensed data periodically for user's area of interest.

7.1 Summary of Contributions

- Performance evaluation for queries and network schemes. We have performance evaluation that consider query characteristics and different network scheme that considers storage scheme and routing protocols. Characteristics of queries and network schemes in wireless sensor networks are studied and evaluated to determine what kind of queries are suitable for what kind of network schemes in terms of different factors that should be considered in designing an application of wireless sensor networks.

- Energy-efficient sectioned tree for query dissemination. On top of MBR spatial indexing, we propose sectioned tree that divides network area into several grid sections within which local trees are built. This saves energy more than MBR indexing only when we disseminate queries to areas of interest. This sectioned tree uses only one path to disseminate queries to one section, which saves more energy than when we use only MBR indexing that may use multiple paths.
- Energy-efficient data gathering tree. We define energy-efficient data gathering tree problem that considers two-way communication that prevents packet collision, overhearing energy consumed by idle nodes, and idle energy consumption by nodes. We prove the EDG problem is NP-complete and propose heuristic algorithms that backtrack one (BT1) or two (BT2) steps back when it creates a tree based on MST. This saves energy when data are gathered and reported to a base station.

7.2 Future Research Direction

We considered homogeneous sensor nodes that perform same tasks throughout our thesis. But, since the real world wants to have many different situations, we need to consider heterogenous sensor nodes that collect different types of data, that have different hardware specification, and that have different architecture, for example. Also we need to handle homogeneous sensor data to eventually provide appropriate information from sensor field to users. Therefore future research can include architectures that consider heterogeneous sensor nodes and data management from sensed data.

REFERENCES

[1] M. Srivastava, "Power considerations for sensor networks," http://ipsn.acm.org/2001/slides/Srivastava.pdf, 2001.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[3] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," *SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 2, pp. 28–36, 2002.

[4] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, "A line in the sand: A wireless sensor network for target detection," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 46, no. 5, pp. 605–634, 2004.

[5] Q. Zhao, A. Swami, and L. Tong, "The interplay between signal processing and netwroking in sensor networks," *IEEE Signal Processing Magazine*, pp. 84–93, July 2006.

[6] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *MOBICOM*, 1999, pp. 263–270.

[7] A. M. Mainwaring, D. E. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring." in *WSNA*, 2002, pp. 88–97.

[8] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. M. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks." *Commun. ACM*, vol. 47, no. 6, pp. 34–40, 2004.

[9] Z. Butler, P. Corke, R. Peterson, and D. Rus, "From robots to animals: Virtual fences for controlling cattle," *International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 485–508, 2006.

[10] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, no. 2, pp. 18–25, 2006.

[11] C. Hartung, R. Han, C. Seielstad, and S. Holbrook, "Firewxnet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments," in *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*. New York, NY, USA: ACM, 2006, pp. 28–41.

[12] D. M. Doolin and N. Sitar, "Wireless sensors for wildfire monitoring," in *Smart Structures and Materials 2005: Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems.*, 2005.

[13] A. Ailamaki, C. Faloutsos, P. S. Fischbeck, M. J. Small, and J. VanBriesen, "An environmental sensor network to determine drinking water quality and security." *SIGMOD Record*, vol. 32, no. 4, pp. 47–52, 2003.

[14] S. Kim, S. Pakzad, D. E. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," in *IPSN*, 2007, pp. 254–263.

[15] M. Li and Y. Liu, "Underground structure monitoring with wireless sensor networks," in *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*. New York, NY, USA: ACM, 2007, pp. 69–78.

[16] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2004, pp. 13–24.

[17] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. I. Corke, "Data collection, storage, and retrieval with an underwater sensor network." in *SenSys*, 2005, pp. 154–165.

[18] V. Chandrasekhar, W. K. Seah, Y. S. Choo, and H. V. Ee, "Localization in underwater sensor networks: survey and challenges," in *WUWNet '06: Proceedings of the 1st ACM international workshop on Underwater networks*. New York, NY, USA: ACM, 2006, pp. 33–40.

[19] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li, "Research challenges and applications for underwater sensor networking," in *WCNC*, vol. 1, 2006, pp. 228–235.

[20] H. Lee, K. Park, B. Lee, J. Choi, and R. Elmasri, "Issues in data fusion for healthcare monitoring," in *to appear in PETRA 08*, 2008.

[21] U. Varshney, "Managing wireless health monitoring for patients with disabilities," *IT Professional*, vol. 8, no. 6, pp. 12–16, 2006.

[22] V. Shnayder, B. rong Chen, K. Lorincz, T. R. F. F. Jones, and M. Welsh, "Sensor networks for medical care," in *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2005, pp. 314–314.

[23] U. Varshney, "Pervasive healthcare and wireless health monitoring," *Mob. Netw. Appl.*, vol. 12, no. 2-3, pp. 113–127, 2007.

[24] J. Chinrungrueng, U. Sunantachaikul, and S. Triamlumlerd, "Smart parking: An application of optical wireless sensor network," in *Applications and the Internet Workshops, SAINT 2007*, 2007, pp. 66–66.

[25] M. Zhang, J. Song, and Y. Zhang, "Three-tiered sensor networks architecture for traffic information monitoring and processing," in *Intelligent Robots and Systems, IROS 2005*, 2005, pp. 2291–2296.

[26] Y. Zhang, X. Huang, and L. Cui, "Lightweight signal processing in sensor node for real-time traffic monitoring," in *ISCIT '07*, 2007, pp. 1407–1412.

[27] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy, "Ultra-low power data storage for sensor networks," in *IPSN '06*. New York, NY, USA: ACM Press, 2006, pp. 374–381.

[28] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks." in *OSDI*, 2002.

[29] P. Bonnet, J. Gehrke, and P. Seshadri, "Querying the physical world," *IEEE Personal Communications*, vol. 7, no. 5, pp. 10–15, October 2000.

[30] N. Sadagopan, B. Krishnamachari, and A. Helmy, "The acquire mechanism for efficient querying in sensor networks," in *Proc. of the first International Workshop on Sensor network Protocol and Applications*, 2003.

[31] O. S. Kaya and S. Baydere, "Response based classification of sensor query networks," in *Proc. of International Conference, APIS '04*, 2004.

[32] X. Li, Y. J. Kim, R. Govindan, and W. Hong, "Multi-dimensional range queries in sensor networks," in *Proc. of the 1st international conference on Embedded networked sensor systems, SenSys '03*, 2003, pp. 63–75.

[33] E. Elnahrawy and B. Nath, "Cleaning and querying noisy sensors," in *Proc. of the 2nd ACM international conference on Wireless sensor networks and applications, WSNA '03*, 2003, pp. 78–87.

[34] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "Ght: a geographic hash table for data-centric storage," in *Proc. of the 1st ACM international workshop on Wireless sensor networks and applications, WSNA '02*, 2002, pp. 78–87.

[35] X. Cheng, B. Narahari, R. Simha, M. X. Cheng, and D. Liu, "Strong minimum energy topology in wireless sensor networks: Np-completeness and heuristics." *IEEE Trans. Mob. Comput.*, vol. 2, no. 3, pp. 248–256, 2003.

[36] T. H. Cormen, C. E. Leiserson, and R. Rivest, *Introduction to Algorithms*, 1990.

[37] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks." in *HICSS*, 2000.

[38] P. Basu and J. Redi, "Effect of overhearing transmissions on energy efficiency in dense sensor networks." in *IPSN*, 2004, pp. 196–204.

[39] W. Ye, J. S. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks." in *INFOCOM*, 2002.

[40] E. Althaus, G. Calinescu, I. I. Mandoiu, S. K. Prasad, N. Tchervenski, and A. Zelikovsky, "Power efficient range assignment for symmetric connectivity in static ad hoc wireless networks." *Wireless Networks*, vol. 12, no. 3, pp. 287–299, 2006.

[41] C. Buragohain, D. Agrawal, and S. Suri, "Power aware routing for sensor databases," in *INFOCOM 2005*, vol. 3, 2005, pp. 1747–1757.

[42] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: A survey," *Wireless Communications*, vol. 11, no. 6, pp. 6–28, 2004.

[43] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM, 1999, pp. 174–185.

[44] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks." in *MOBICOM*, 2000, pp. 56–67.

[45] D. Braginsky and D. Estrin, "Rumor routing algorthim for sensor networks," in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM, 2002, pp. 22–31.

[46] A. Manjeshwar and D. P. Agrawal, "Teen: A routing protocol for enhanced efficiency in wireless sensor networks." in *IPDPS*, 2001, p. 189.

[47] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, "Ttdd: two-tier data dissemination in large-scale wireless sensor networks," *Wirel. Netw.*, vol. 11, no. 1-2, pp. 161–175, 2005.

[48] J. N. Al-Karaki, R. Ul-Mustafa, and A. E. Kamal, "Data aggregation in wireless sensor networks - exact and approximate algorithms," in *High Performance Switching and Routing*, 2004, pp. 241–245.

[49] B. Karp and H. T. Kung, "Gpsr: greedy perimeter stateless routing for wireless networks," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000, pp. 243–254.

[50] D. E. Y. Yu and R. Govindan, "Geographical and energy-aware routing: A recursive data dissemination protocol for wireless sensor networks," UCLA Computer Science Department, Technical Report UCLA-CSD TR-01-0023, May 2001.

[51] S. Wu and K. S. Candan, "Gper: Geographic power efficient routing in sensor networks." in *ICNP*, 2004, pp. 161–172.

[52] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-centric storage in sensornets with ght, a geographic hash table," *Mobile Networks and Applications*, vol. 8, no. 4, pp. 427–442, 2003.

[53] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks." in *SIGMOD Conference*, 2003, pp. 491–502.

[54] A. Soheili, V. Kalogeraki, and D. Gunopulos, "Spatial queries in sensor networks." in *GIS*, 2005, pp. 61–70.

[55] A. Coman, M. A. Nascimento, and J. Sander, "Exploiting redundancy in sensor networks for energy efficient processing of spatiotemporal region queries." in *CIKM*, 2005, pp. 187–194.

[56] J. Beaver, M. A. Sharaf, A. Labrinidis, and P. K. Chrysanthis, "Location-aware routing for data aggregation for sensor networks," in *Proceedings of GeoSensor Networks Workshop*, 2003. [Online]. Available: citeseer.ist.psu.edu/beaver03locationaware.html

[57] H. Yang, F. Ye, and B. Sikdar, "A dynamic query-tree energy balancing protocol for sensor networks," in *WCNC 2004: Wireless Communications and Networking Conference*, 2004, pp. 1715–1720.

[58] M. Cagalj, J.-P. Hubaux, and C. C. Enz, "Energy-efficient broadcasting in all-wireless networks." *Wireless Networks*, vol. 11, no. 1-2, pp. 177–188, 2005.

[59] M. Khan, G. Pandurangan, and B. Bhargava, "Energy-efficient routing schemes for wireless sensor networks," Perdue University, Computer Science, Technical Report CSD TR-03-013, 2003.

[60] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," in *INFOCOM (2)*, 2000, pp. 585–594.

[61] Y.-W. Hong and A. Scaglione, "Energy-efficient broadcasting with cooperative transmissions in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 10, pp. 2844–2855, 2006.

[62] W. Liang and Y. Liu, "Online data gathering for maximizing network lifetime in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 1, pp. 2–11, 2007.

[63] L. M. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc, "Power consumption in packet radio networks (extended abstract)." in *STACS*, 1997, pp. 363–374.

[64] A. E. F. Clementi, P. Penna, and R. Silvestri, "Hardness results for the power range assignmet problem in packet radio networks." in *RANDOM-APPROX*, 1999, pp. 197–208.

[65] R. C. Prim, "Shortest connection networks and some generalizations," *Bell System Tech. J.*, vol. 36, pp. 1389–1401, 1957.

[66] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *American Mathematical Society*, vol. 7, no. 1, pp. 48–50, Feb. 1956.

[67] S. C. Narula and C. A. Ho, "Degree-constrained minimum spanning tree," *Computers and operations Research*, vol. 7, no. 4, pp. 239–249, 1980.

[68] M. R. Garey and D. S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman, January 1979.

[69] C. H. Papadimitriou, "The complexity of the capacitated tree problem," *Networks*, vol. 8, pp. 217–230, 1978.

[70] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*. Springer, 2008.

[71] M. Obitko, "Introduction to genetic algorithms - tutorial with interactive java applets," http://www.obitko.com/tutorials/genetic-algorithms/index.php, 1998.

[72] C. R. Reeves and J. E. Rowe, *Genetic Algorithms - Principles and Perspectives*. 101 Philip Dr., Assinippi Park, Norwell, Massachusetts 02061 USA: Kluwer Academic Publishers, 2003.

[73] T. Back, *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[74] G. R. Raidl and B. A. Julstrom, "Edge sets: An effective evolutionary coding of spanning trees," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 3, pp. 225–239, 2003.

[75] H. Chou, G. Premkumar, and C.-H. Chu, "Genetic algorithms for communications network design - an empiricalstudy of the factors that influence performance," *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 3, pp. 236–249, 2001.

[76] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer, 2003.

[77] G. R. Raidl, "An efficient evolutionary algorithm for the degree-constrained minimum spanning tree problem," vol. 1, 2000, pp. 104–111.

[78] S. Kirkpatrick, D. G. Jr., and M. P. Vecchi, "Optimization by simmulated annealing." *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[79] S. R. Schmitt, "Simulated annealing demonstration," http://home.att.net/∼srschmitt/sa_demo/SA-demo.html, 2004.

[80] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*, 2nd ed. Springer-Verlag Berlin Heidelberg New York, 2004.

[81] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks." in *MOBICOM*, 2002, pp. 148–159.

[82] C. Intanagonwiwat, R. Govindan, D. Estrin, J. S. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking." *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 2–16, 2003.

[83] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari, "Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks," in *Proc. of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[84] S. Coleri, A. Puri, and P. Varaiya, "Power efficient system for sensor networks." in *ISCC*, 2003, pp. 837–842.

[85] K. Park and R. Elmasri, "Query classification and storage evaluation in wireless sensor networks." in *ICDE Workshops (NetDB 06)*, 2006, p. 35.

[86] J. Carle and D. Simplot, "Energy-efficient area monitoring for sensor networks." *IEEE Computer*, vol. 37, no. 2, pp. 40–46, 2004.

[87] M. Stemm and R. H. Katz, "Measuring and reducing energy consumption of network interfaces in hand-held devices," *IEICE Transactions on Communications*, vol. E80-B, no. 8, pp. 1125–31, 1997. [Online]. Available: citeseer.ist.psu.edu/stemm97measuring.html

[88] O. Kasten, "Energy consumption," http://www.inf.ethz.ch/personal/kasten/research/ bathtub/energy_consumption.html, 2001.

[89] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *Wireless Networks*, vol. 8, no. 5, pp. 481–494, 2002.

BIOGRAPHICAL STATEMENT

Kyungseo Park was born in Seoul, Korea. He received his B.E. degree in Electronic Engineering from Hongik University, Korea, in 1995, his M.S. and Ph.D. degrees from The University of Texas at Arlington in 2003 and 2008, respectively, all in Computer Science and Engineering. From 1995 to 2000, he was in Samsung Semiconductor, Kiheung, Korea, where he worked as a maintenance and service engineer for photolithographic equipment. His current research interest is in the area of data management and communication in wireless sensor networks.