Power-aware routing in wireless sensor networks

Zha, Wei

2013

Zha, W. (2013). Power-aware routing in wireless sensor networks. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/54843>

<https://doi.org/10.32657/10356/54843>

NANYANG TECHNOLOGICAL UNIVERSITY

Power-aware Routing in Wireless Sensor Networks

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

by

Zha Wei

Supervisor: Ng Wee Keong

School of Computer Engineering

2013

Abstract

Wireless sensor networks have drawn a lot of attention in recent years. Due to the advantages of flexibility and cheap cost, wireless sensor network applications have been widely used for both civil and military purposes. However, constrained by hardware, sensor nodes usually have limited energy supply. All wireless sensor network applications have to take energy issues into consideration. In this thesis, we explore energy efficiency in wireless sensor networks and focus on energy efficient routing protocols. Two kinds of routing protocols, spatial index based routing protocols and geographic based routing protocols, are studied. Spatial index based routing protocols are able to inquire a selected area without flooding the whole network. The drawbacks are the high cost of maintenance and the lack of scalability. Strategies which prolong sensor network lifespan by reducing maintenance cost is studied. Geographic based routing protocols, on the contrary, are quite scalable because the network is organized in a stateless manner. However, they are all designed for stationary networks. Releasing this assumption may fail to guarantee delivery. The deep causes of delivery failure in duty-cycling sensor networks are fundamentally investigated and formally analyzed. A general algorithm component is proposed to serve existing geographic routing algorithms to guarantee the success of packet delivery in duty-cycling sensor networks. Future work will continue studying practical implementations of geographic routing and explore some new areas, such as energy usage prediction and privacy preserving in wireless sensor networks.

Acknowledgments

I would like to express my gratitude to my supervisor, **Associate Professor Ng Wee Keong**, for his invaluable guidance, advice, encouragement and sharing of his knowledge. It would not have been possible to write this thesis without his support. Because of him, I started to know the world of research and do the research seriously as a part of my life. I have learned how to think systematically, deeply, and carefully. I have also learned how to write and present my work properly and clearly. I am grateful to him for being my mentor.

Besides my supervisor, I would like to thank: **Nanyang Assistant Professor. Li Mo**, for his encouragement and patience in guiding me. His insightful comments have brought my research one step closer to world class research.

In my daily work, I have been blessed with a friendly and cheerful group of students and staff: **Lua Ruiping, Goh Weihan, Zhang Ran** and **Vincent Tan**. They generously share with me all their knowledge and skills.

I thank my fellow students, **Dr. Han Shuguo, Dr. Zhu Linhong, Wan Li** and **Zhai Ke**. They keep offering me help all the time even after their graduation.

My thanks also go to people in Parallel and Distributed Computing Centre, **Dr. Liu Fei**, **Dr. Luo Linbo**, **Dr. Zhang Huiliang**, **Dr. Lee KenYoong**, **Dr. Wong Lipei**, **Wen Hongkai**, **Hao Jianan**, **Fan Hongfei**, **Mao Yuqing**, **Zhang Lu**, **Zheng Yang** and **Dong Jie**. Years of collaboration have strengthen our bonds of friendship. I thank them for all the fun we have had in the last few years.

Besides them, I also want to thank **Loo Kian Hock**, for his assistance in providing the administration and technical support during my stay at the Data-Intensive Scalable Computing Lab.

Last but not least, I must sincerely thank **my parents** and **my wife** for their love, care and support throughout my life. Without them, this work would not be possible.

Thank you all.

Zha Wei

19, August, 2013

Contents

Abstract	i
Acknowledgments	ii
List of Figures	vii
1 Introduction	1
1.1 Sensor Network Applications	1
1.2 Challenges and Issues in Sensor Networks	9
1.3 Research Objectives	16
1.4 Report Organization	18
2 Background Knowledge	19
2.1 Overview	19
2.2 Spatial Index Based Approach	19
2.2.1 Peer-Tree	23
2.2.2 Spatial Index Tree	25
2.2.3 OMSI-Tree	28
2.2.4 Subarea Tree Routing	30
2.2.5 Sectioned Tree	33
2.2.6 Distributed Quad-Tree	35
2.2.7 Quadtree-based Data Dissemination	37
2.2.8 Conclusion	39

2.3	Geographic Routing Based Approach	41
2.3.1	Greedy Routing	41
2.3.2	Face Routing	41
3	Multi-parents Approach	49
3.1	Multi-parent Methods	50
3.1.1	OMSI-Tree	50
3.1.2	Nearest-Parent Tree	51
3.1.3	Hybrid Structure	53
3.1.4	Multi-Parent Methods	54
3.2	Experimental Results	59
3.3	Summary	62
4	Virtual Node Approach	63
4.1	Virtual Node Methods	66
4.1.1	Node Redundancy	66
4.1.2	Virtual Node	67
4.1.3	Virtual Node Approach for Physical Redundancy	68
4.1.4	Virtual Node Approach for Logical Redundancy	73
4.1.5	Maximal Size of Virtual Node	75
4.2	Experimental Results	79
4.3	Summary	85
5	Geographic Routing in Duty-Cycling Sensor Networks	86
5.1	Duty-cycling Sensor Networks	87
5.2	Motivations	88
5.3	Analysis of Delivery Failure	91

5.3.1	Atomic Network Topology Changes due to Node Sleep and Wake-up	91
5.3.2	Delivery Failure due to Packet Leaving Major Face	93
5.3.3	Delivery failure while packet is still on major face	96
5.4	Algorithm Design	97
5.4.1	Algorithm Properties	101
5.4.2	Delivery Guarantee	102
5.5	Experimental Results	104
5.5.1	Improvement of Proposed Algorithm Component in Duty-cycling Sensor Network	105
5.5.2	Loops in Duty-cycling Sensor Network	106
5.5.3	Overheads	106
5.6	Summary	107
6	Conclusion and Future Work	115
6.1	Conclusion	115
6.2	Future Work	116
	References	120

List of Figures

1.1	Example of sensor network architecture	2
1.2	Wireless sensors in support of base protection	3
1.3	System architecture for habitat monitoring	5
1.4	System architecture for environment observation in CORIE	7
1.5	System architecture of CodeBlue	8
2.1	Example of R-tree	21
2.2	Quad-tree	23
2.3	SPIX tree	25
2.4	Effect of eliminating thin rectangles	27
2.5	Effect of energy optimization phase	28
2.6	Example of OMSI-tree	29
2.7	Construction of STR	32
2.8	Naive routing tree versus Sectioned tree	33
2.9	Distributed quad-tree	35
2.10	Partition of QDD	38
2.11	Greedy routing	40
2.12	Greedy routing failure	41
2.13	Right hand rule	42
2.14	Gabriel graph and relative neighborhood graph	44
2.15	Face routing failure	46

2.16	Face routing part of existing geographic routing algorithms	46
3.1	OMSI-tree	51
3.2	Nearest parent tree	52
3.3	Hybrid tree	53
3.4	Experiment results of (a) OMSI-tree, (b) Nearest parent tree and (c) Hybrid tree with radio radius = 2	55
3.5	Experiment results of (a) OMSI-tree, (b) Nearest parent tree and (c) Hybrid tree with radio radius = 4	56
3.6	Experiment results of (a) OMSI-tree, (b) Nearest parent tree and (c) Hybrid tree with radio radius = 6	57
4.1	Virtual grid and virtual nodes on sensor network	64
4.2	Possible node location inside an atomic-cell	69
4.3	Virtual node approach for logical redundancy	74
4.4	Connection between virtual nodes	75
4.5	Maximal size of virtual nodes	76
4.6	Performance without virtual nodes	79
4.7	Performance of virtual nodes	80
4.8	Average latency with different virtual node size and without virtual node	82
5.1	Observation of delivery failures	89
5.2	Example of face merge and split	92
5.3	Delivery failure caused by face split and merge	108
5.4	Delivery failure caused by face merge and split	109
5.5	Possible new major face positions due to topology changes	110
5.6	Successful packet delivery of GPSR with our algorithm	111

5.7	Comparison between GPSR solely apply and with our algorithm in success rate. N is the number of source-destination pairs has been executed	112
5.8	Loops recorded under 80% active duty-cycle with 100 source-destination pairs. Arrows indicate packet delivery direction.	113
5.9	The number of loops at each time slot	113
5.10	Memory occupancy with different network size	114

Chapter 1

Introduction

A wireless sensor network consists of dozens to thousands of sensor nodes that are powered by batteries of limited lifespan. Sensor nodes are usually spatially deployed in a monitored area of limited accessibility to human beings or in an area that is not suitable for humans to stay, such as the virgin forest, nuclear stations and so on. Deployed sensor nodes work cooperatively to monitor the environment. Currently, a sensor node is able to detect temperature, humidity, lighting, object movement and so on. Sensor readings are gathered by a sink node and sent to a remote server via Internet or using satellites. Users access the information of the monitored area through that server. Figure 1.1 describes a typical sensor network's architecture. Sensor nodes pass their readings to a sink node in a multi-hop manner. Upon receiving the sensor data, the sink node sends them to a remote server via the Internet or using satellites for off-line computation and analysis. Users may then take actions based on the analyzed results.

1.1 Sensor Network Applications

Sensor networks are widely used in many applications as they can be deployed in many environments that are not suitable for human beings. We classify these applications into two major domains: military applications and civil applications.

Figure 1.1: Example of sensor network architecture

Military Applications

Sensor networks used for military purposes can help with the detection of enemy movements, identification of enemy forces, analysis of their movements and progress, as well as monitoring of militant activities in remote areas of specific interest and in force protection.

- **Base Protection.** Figure 1.2 shows a case of base protection in military application. Having situated the headquarters in an area of active engagement, it is essential to prevent the base from attacks. The surrounding terrain may be undulating or mountainous and potentially be obscured in trees and vegetation. Attacks could come in the form of militant groups on foot or with motor vehicles. In order to facilitate an early detection, the perimeter of protection must cover a belt around the camp of up to 4 km, which in practice has ranges of up to 10 km. Detection may be needed throughout the entire range whilst identification may only be required within a belt of around 1 to 2 km around the base.

Figure 1.2: Wireless sensors in support of base protection

- **Battle Field Monitoring.** A dynamically placed intrusion sensor network is another kind of military application. In this case, if an army force has cleared a building but cannot afford to leave people behind to safeguard it, they place sensor nodes in the building in order to notify them if anyone enters the cleared building after they have left. This is quite essential when the battlefield is in the city.
- **Other Military Systems with Sensor Networks.** Other sensor network applications in the military can be found in C4ISRT (command, control, communications, computing, intelligence, surveillance, reconnaissance and targeting) systems [50, 1] and NBC (nuclear, biological and chemical) systems.

Civil Applications

Major civil applications can be classified by their purposes: habitat monitoring, environment observation, health monitoring, structure health monitoring and so on. Figure 1.3 shows an architecture for habitat monitoring. The lowest level of the sensing application is provided by autonomous sensor nodes. Individual sensor nodes communicate and coordinate with one another. Each sensor node is equipped with a communication module that broadcasts sensor data to other sensor nodes and base station. The base station stores data into the data server through internet and allows end-users to access the data. Environment Observation and Forecasting System consists of three parts: sensor stations, a distribution network and a centralized processing center. It is mainly used to monitor, model and forecast physical phenomena. Sensor networks used by health applications include glucose level monitoring, organ monitoring, cancer detection and general health monitoring. Biomedical applications with small embedded wired and wireless sensors are promising in the future. Structure health monitoring process involves the observation of a system over time using periodically sampled dynamic response measurements from an array of sensors. Sensor nodes are able to detect damage, localize damage, estimate the extent of damage and predict residual life of a structure [49]. In addition, ambient computing gives us the idea of smart home/office environment which is entirely based on sensor networks. For instance, the Smart Kindergarten [43] consists of a wireless sensor network for early childhood education. Some of the projects are shown as follows:

- In Great Duck Island (GDI) system [40], people are primarily interested in three major issues in monitoring the Leach's Storm Petrel:

Figure 1.3: System architecture for habitat monitoring

- (i) What is the usage pattern of nesting burrows over the 24 – 72 hour cycle when one or both members of a breeding pair may alternate incubation duties with feeding at sea?
- (ii) What changes can be observed in the burrow and surface environmental parameters during the course of the approximately seven-months breeding season (April-October)?
- (iii) What are the differences in the micro-environments with and without large numbers of nesting petrels?

The authors adopted the system architecture for habitat monitoring to obtain all the information they need.

- PODS project [4] is conducted by University of Hawaii to investigate why endangered species of plants will grow in one area but not in the neighboring

areas. Environmental sensors, called Pods, are camouflaged and deployed in Hawaii Volcanoes National Park. Each pod is equipped with a computer, a radio transceiver and a high resolution digital camera. Pods communicate with each other via wireless link. Bluetooth and 802.11b are chosen as MAC protocols.

Energy efficiency has been specified as one of the design goals. The routing protocol called Multi-Path On-demand Routing (MOR) was developed to meet this design objective. There are two types of data collected: weather data, collected every ten minutes and image data, collected once per hour. All these collected data are accessible from a server in University of Hawaii at Manoa via the Internet.

There are some other works which also concentrate on habitat monitoring by using wireless sensor networks [10, 55, 56].

- CORIE [13] is a prototype of environment observation and forecasting system for Columbia river. This prototype consists of 24 sensor stations, both stationary stations and mobile stations. Stationary stations are powered by power grids and mobile stations are equipped with solar panels to recharge their energy. Therefore, the energy issue does not get highlighted in this project.

Each station is equipped with variable combinations of in-situ sensors to measure one or more physical properties of water or atmosphere. Water temperature, salinity, and water levels are measured by most stations. Profiles of velocity and acoustic backscatter are measured by three stations. The sensor data are transmitted via wireless link to the stations. The stations further forward the data to a centralized server for computation.

Figure 1.4: System architecture for environment observation in CORIE

Components of the CORIE modeling system are shown in Figure 1.4. All the environment properties are observed by sensors as shown in the upper part of Figure 1.4. The measurement methods include detection by a sensor network, remote sensing and radar doppler. All the information is gathered at a central server for data assimilation and forecasting.

- Most people agree that ALERT (Automated Local Evaluation in Real-Time) [27] is the first well-known wireless sensor network which was deployed by the National Weather Service in the real world. Real-time rainfall and water level information is provided by ALERT to evaluate potential flooding. Water level sensor, temperature sensor and wind sensor are equipped at each sensor node. Communication is transferred via radio from sensors to base station. At base

Figure 1.5: System architecture of CodeBlue

station, all these data are processed by a Flood Forecast Model to forecast potential flooding and warnings are issued automatically. More examples of environment observation can be found in these works [5, 53].

- CodeBlue [41] is an ad hoc sensor network infrastructure for emergency medical care proposed by Harvard University. By integrating low-power, wireless vital sign sensors, PDAs, and PC-class systems, CodeBlue allows close monitoring of patients and seamless transfer of patient data among caregivers to facilitate efficient allocation of hospital resources. The architecture of CodeBlue is illustrated by Figure 1.5.

As the world's population ages, the elderly suffering from diseases will increase. In-home and nursing-home ubiquitous networks may assist residents and their caregivers by providing continuous medical monitoring, memory enhancement, control of home appliances, medical data access, and emergency communica-

tion. Doctors can more efficiently monitor their patients with the assistance of sensor networks as well [9, 8]. For instance, caregivers can be made aware of a patient's blood pressure and be notified when the blood pressure gets high.

- Wind and Structural Health Monitoring System(WASHMS) [2]. WASHMS for bridges in Hong Kong is a sophisticated bridge monitoring system, costing USD 1.3 million. It is used by the Hong Kong Highways Department to ensure road users' comfort and safety of the Tsing Ma, Ting Kau, and Kap Shui Mun bridges that run between Hong Kong and Hong Kong International Airport.

WASHMS consists of four parts: sensory systems, data acquisition systems, local centralised computer systems and global central computer system. The sensory system consists of approximately 900 sensors including 350 sensors on the Tsing Ma bridge, 350 on Ting Kau and 200 on Kap Shui Mun. Sensors include accelerometers, strain gauges, displacement transducers, level sensing stations, anemometers, temperature sensors and dynamic weight-in-motion sensors. The structural behaviors of the bridges are measured twenty four seven.

In addition, ambient computing brings the ideas of smart home/office environment which are absolutely based on sensor networks. For instance, the Smart Kindergarten [43] consists of a wireless sensor network for early childhood education.

1.2 Challenges and Issues in Sensor Networks

Before putting sensor nodes into both military and civil usages, there are various challenges and issues that need to be addressed.

Physical size of Sensors

Although sensor nodes may be carried by vehicles or aircrafts to the area of deployment, the physical size of sensor nodes are expected to not be too big. A match-box size is preferred. Considering some nodes may need to be equipped with an antenna and powered by a super battery, a shoe-box size is also acceptable. Especially, for military purposes, it would be beneficial if the nodes were covert in appearance with a small electromagnetic emission pattern so as to remain hidden from potential adversaries. That is why match-box size sensors are preferred.

However, larger size of sensor nodes can be equipped with more batteries which means that the sensor network can be operated for a longer time. Tradeoff between sensor nodes' size and the operating time must be made.

Self-configuration after Deployment

Since sensor nodes may be deployed to the monitored area by aircraft or helicopter, they must be able to rapidly identify their neighbours within communication range and configure themselves into an ad hoc network.

Sensor nodes are deployed without the knowledge of other nodes' location information. A given sensor's view is limited. To rapidly organize sensor nodes into an ad hoc network is quite challenging.

Nodes may also be destroyed or may fail during operation. The other challenge is that the network should be able to reconfigure itself and no manual intervention should be involved.

Scalability

In civil usage, sensor networks are mainly used to monitor a small area. For instance, smart home project only requires sensor nodes to monitor a room size where several

sensor nodes should be enough. While in military usage, the network is usually expected to cover an area from 5 to $20km^2$ and consist of hundreds or thousands of sensor nodes.

The sensor network routing scheme must be able to work with this huge number of sensor nodes. Further more, the sensor network may be required to work without the pre-knowledge of the network scale.

Information Flows

Generally, one-way communication is sufficient for observation purpose in a sensor network. In military usage, all the sensor nodes pass detected information to the sink node, from where a commander could update the battlefield situation. In civil usage, the temperature of the monitored area can be passed to a sink node by one-way communication. Thus one-way communication is sufficient for most monitoring cases.

While in some cases, two-way communication is necessary. For instance, when a commander wants to change the orientation of the camera on a sensor node or a user wants to execute a spatial query. Two-way communication brings the interaction between users and sensor nodes. However, two-way communication also brings the secure issues. In addition, synchronization among sensor nodes may be required in two-way communication.

Lifespan of Network

The expected lifespan of sensor network is application based. In civil applications, especially in environment observation and habitat monitoring, sensor networks are usually expected to be operated for months and years. For example, in GDI project, the sensor network works from spring to mid-October of 2002. While in military usages, sensor networks are usually not required to be in operation for a long period.

For instance, the network deployed for the purpose of detection is not necessarily required to be functional after the force starts attacking. In addition, some networks are switched on only during the night.

Estimation of sensor network's lifespan should be made before deployment, so that the sensor network can schedule the energy consumption properly. The estimation should consider the factors from all aspects to make sure that the sensor network fulfills the duration requirement. This is one of the challenges in sensor network design.

Spatial Query

Spatial query allows the network to be queried by location rather than by an attribute. This is especially useful in both military and civil applications. For instance, to check if any vehicle has entered the monitored area, the army force can query the sensor nodes deployed beside the road. The sensor nodes deployed on the top of the mountain or in the forest would not be helpful in this case. In civil applications, spatial query is useful as well. Forest manager may keep querying the regions with unusual temperature to forecast a forest fire. Spatial query gives the possibility to query a specific area instead of the entire network.

The major challenge to support spatial query is that there is no global address scheme in sensor networks. Each node is aware of its neighbors only. Spatial query cannot be probably delivered to sensor nodes inside the query area without a global scheme.

Data Type

Generally speaking, all the sensor readings can be represented by a few Bytes. For example, dimensionality and longitude would be enough to locate an object. In

addition, environment temperature, object velocity and other values can all be represented by a few Bytes.

However, real-time images and video data are likely to be requested in the near future. New data types, such as image and video are expected to be supported in sensor networks.

Routing Protocols

How the information is delivered from sensor nodes to sink nodes inside sensor networks is controlled by routing protocols. For instance, whether the information is delivered in a single-hop manner or in a multi-hop manner and how the information is routed to the destination in a multi-hop manner. Routing protocols should also consider sensor network self-configuration and concern about traffic latency, networking overheads, energy efficiency, the speed of network recovery in case of failures, traffic assurance and so on. Three major classifications of routing protocols for energy-efficient wireless sensor networks have been identified:

- **Flooding based** [24, 59, 42, 7, 26]. This is a direct implementation of wireless protocols. Whenever a node needs to send out a packet, it broadcasts to the whole network. Each node forwards that packet at least once. This is inefficient due to the involvement of unnecessary nodes. In addition, network congestion might be caused if multiple nodes forward the packet concurrently. It can hardly work in large scale wireless sensor network.
- **Hierarchical/node-centric** [23, 36, 35]. Most earlier routing protocols follow this approach. These protocols aim at clustering the nodes so that cluster heads can perform some in-network computation, such as aggregation, which reduces the amount of data to be transmitted and conserves energy. However, their

routing tables may take time to converge (i.e., choose the most appropriate route) if frequent network topology changes occur (which can happen if nodes can transition into suspend mode to conserve energy).

- **Geographical Routing** [48, 57, 61]. This routing class is based on the exact (GPS) or relative (triangulation, analysis of neighbor dependencies) position of the single node. In general, a geographic routing algorithm consists of two stages: greedy routing and face routing. In greedy routing, each node keeps forwarding packet to one of its neighbors, which is closest to final destination, to ensure the packets approach the destination after sufficient forwardings. However, in some situations, greedy routing may lead packets to a local minimum node and packets may get stuck at that node with all neighboring nodes which are not any closer to the final destination. If such a situation occurs, face routing will be invoked to route packets out of the local minimum node.

Energy Efficiency

Sensor nodes are powered by battery with a limited lifespan. Energy is critical to keep a sensor network functional. Usually, it is hard or impossible to replace or recharge the equipped batteries, which means sensor networks must efficiently use the limited energy especially for the networks which are designed for a long duration of usage. Energy efficiency can be achieved from both hardware and application views. Recent hardware advantages bring us low power consumption sensor nodes with storage and processing capabilities. Sensor nodes are able to work longer but consume less energy. With the capabilities of storage and computation, energy efficient strategies are also possible to be carried out from application level.

Traditional sensor network applications periodically collect the detected information from a sink node to a database and apply off-line computing for analysis and

prediction. Usually, a sink node is assumed to be a super node with an unlimited power supply, while the rest of the sensor nodes are still powered by battery with a limited lifespan. It is expensive to recharge or replace the battery of sensor nodes. Among all the operations in a sensor node, data communication consumes the most energy. For instance, energy consumption of one message sending operation is at least 1,000 times more expensive than a local operation (e.g., sensing data). Hence, power consumption and communication are major issues in wireless sensor networks, especially in civil applications which usually require that the sensor networks be operated for a long time. There is a deep relation between energy consumption and communication traffics. Reducing unnecessary communication among sensor nodes is one of the major approaches to prolong the lifespan of sensor networks.

Specifically, energy cost of sending one message over a distance of d is

$$e = \kappa d^c$$

where κ and c are constants and c is usually between 2 and 4. It is clear that multi-hop transmission is much more energy efficient than single-hop transmission. The less distance per hop, the less energy is incurred. Figure 1.1 shows that messages in the sensor network are passing to sink node in a multi-hop manner. Suppose node B needs to send a message to node D . According to energy consumption equation, in a single-hop manner (we assume $c = 2$ to ease our analyzing), the energy cost is:

$$e_s = \kappa(d_1 + d_2)^2 = \kappa(d_1^2 + d_2^2 + 2 \times d_1 d_2)$$

While in multi-hop manner, the energy cost is

$$e_m = \kappa(d_1^2 + d_2^2)$$

Energy conserved in multi-hop manner is:

$$e_s - e_m = \kappa(d_1^2 + d_2^2 + 2 \times d_1 d_2) - \kappa(d_1^2 + d_2^2) = 2 \times \kappa d_1 d_2$$

We note that more energy could be conserved if c is bigger or more hops are involved in message delivery.

1.3 Research Objectives

Wireless sensor networks are built for various purposes. They may have different requirements such as QoS (Quality of Service), scalability, fidelity, data throughput, etc. But they all face the fact that sensor nodes need to communicate with each other to keep the network functional. Communication cost in wireless sensor network is not trivial since sensor nodes are powered by battery with limited energy supply. Therefore, all wireless sensor network applications have to carefully choose a routing protocol where the packet can be delivered with low energy consumption while fulfilling the application's objective.

Our research objective is to explore exiting routing algorithms and design new energy efficient routing algorithms for wireless sensor networks. In this thesis, we mainly explore two kinds of routing algorithms: spatial index based routing algorithms and geographic routing algorithms.

Spatial index is the major approach to support spatial query in wireless sensor networks. It was first proposed in spatial database to index spatial objects. Some well-known spatial index structures such as R-tree [22], R+tree [51] and R*tree [3] are widely studied in the field of spatial databases. Various works have been done to study how to extend spatial index to sensor networks to support spatial queries. Generally, spatial indices adapted to sensor networks adopt a hierarchical structure where sensor nodes are in a parent-child relationship. By broadcasting its ID and location information, a sensor node is able to know other sensor nodes apart from just its neighbors. Each parent node maintains its offspring minimum bounding region (MBR) information. Referring to the MBR information, a child node can be

located with a few hops instead of flooding the whole network. The query could be prevented from flooding over the sensor network and be sent to the relevant node(s) only. Energy efficiency is achieved in this way.

However, all these spatial index strategies face the same problem of maintaining spatial indices. Due to the hierarchical structure, child nodes must be accessed from parent node. Any parent nodes' failure may lead to the offspring nodes being disconnected and inaccessible even if they are still in good health. Disconnected offspring may join the network by rebuilding the spatial index. Each spatial index rebuilding phase leads to an update of the whole sensor network which is not energy efficient and not suitable for a large scale network. In this work, we will design routing protocols to maximally avoid the expensive rebuilding phase.

Geographic routing algorithms for wireless sensor networks guarantee the success of packet delivery from the data source to the destination with sensors' location information. Unlike spatial index based routing algorithm, sensor nodes in geographic routing algorithms are organized in a stateless manner without parent-child relationship.

In general, a geographic routing algorithm consists of two phases: greedy routing and face routing. In greedy routing, each node keeps forwarding packet to one of its neighbors, which is closest to the final destination, to ensure packets to approach the destination after sufficient forwardings. However, in some situations, greedy routing may lead the packets to a local minimum node and packets may get stuck at that node with all neighboring nodes which are not any closer to the final destination. If such a situation occurs, face routing will be invoked to route packets out of the local minimum node. In face routing, a packet usually traverses a series of faces. In each face, if the packet meets a node, which has an edge that intersects with the "virtual line" from the source to the destination, it stops traversing the current face

and switches to the next one. Such an operation is referred to as face changing and it is repeated until the packets reach the destination or get back to the greedy routing state.

There have been intensive studies on developing efficiency and reliable face routing approaches; nevertheless, though most face routing protocols guarantee the success of packet delivery, they implicitly require sensor nodes to always be active. In this work, we will design geographic routing algorithms which work with duty-cycling sensor networks.

1.4 Report Organization

This report is organized as follows: some general background knowledge about spatial index and geographic routing algorithms on sensor networks is introduced in Chapter 2. In Chapter 3, we propose a strategy of multiple return-paths to share the parent nodes' workload. In Chapter 4, we move our scenario to a high-density sensor network. By classifying redundancy into physical redundancy and logical redundancy, we propose the concept of virtual nodes, to group together a set of redundant sensor nodes so that only one node needs to be active while the rest are in the sleeping mode. We move our focus to geographic routing algorithms in Chapter 5. We analyze how traditional geographic routing algorithms fail to guarantee delivery in duty cycling sensor networks. Then a routing algorithm which guarantees delivery under duty cycling sensor networks is introduced. Finally, Chapter 6 presents conclusions for the report and identifies potential research issues that require further study in the next phase.

Chapter 2

Background Knowledge

2.1 Overview

Energy-aware routing protocols can be generally classified into two kinds: spatial index based routing and geographic routing (also known as stateless routing). In spatial index based approach, wireless sensor network is usually organized in a tree structure. Each node can be accurately located by a unique path. In geographic routing, the path to destination is not fixed. Any node between source and destination could be an intermediate node. Spatial index based routing can provide the optimal routing path while geographic routing is more scalable.

2.2 Spatial Index Based Approach

Spatial index was first proposed by database community to handle spatial objects. Traditional index methods are not well suited to data objects of non-zero size located in multi-dimensional spaces.

R-tree [22] was proposed to handle spatial data efficiently. R-tree is a height-balanced tree similar to B-tree, but with index records in its leaf nodes containing pointers to data objects. Leaf node in R-tree maintains the index in the form of $(I, \textit{tuple-identifier})$ where $\textit{tuple-identifier}$ is the spatial object and I is the minimum

bounding region (MBR) of that spatial object. Non-leaf node contains entries in the form of $(I, \textit{child-pointer})$ where *child-pointer* is the address of a lower node in the R-tree and I covers all rectangles in the lower node's entries.

Let M be the maximum number of children that one node could have and let $m \leq M/2$ specify the minimum number of children that a node can have. An R-tree satisfies the following properties:

- All the leaf nodes contain between m and M index records.
- Each leaf node contains an index in the form of $(I, \textit{tuple-identifier})$, where *tuple-identifier* represents the spatial object in n-dimensional space and I is the smallest rectangle that spatially contains that spatial object.
- Each parent node has children between m and M unless it is the root.
- Each non-leaf node contains an index in the form of $(I, \textit{child-pointer})$, where *child-pointer* points to its child nodes and I is the smallest rectangle that spatially contains the rectangles in the child nodes.
- Root node has at least two children as long as it is not a leaf.
- All the leaves stay at the same level.
- The height of an R-tree is at most $\lceil \log_m N \rceil - 1$, where N is the number of spatial objects it contains.

The author also proposed several operations in order to maintain R-tree, including: Searching, Insertion, Deletion, Updating and Node Splitting. All these operations guarantee that R-tree is scalable and adaptive.

An example of R-tree is shown in Figure 2.1. As we can see, each rectangle represents a group of objects and each black dot represents one object. To access

Figure 2.1: Example of R-tree

object A , $R1$ just needs to check if A falls inside its MBR. If so, $R1$ forwards query to its sub-MBR. In this case, $R3$ will match and access object A .

We noticed that only leaves in R-tree point to the object. Unlike the spatial database, sensor nodes inside the sensor network do not have an overview of the whole network. They know their one-hop neighbors only. This challenge prevents R-tree from being directly used in sensor networks. Soheili *et al.* [52] systematically described how to build a spatial index in sensor networks by using a R-tree like approach as well as many other R-tree like approaches.

Quad-tree [15] is another tree structure that can be used to index spatial objects. Particularly, quad-tree is often used to recursively partition a two dimensional space into four quadrants or regions.

According to the type of data they represent, quad-trees may be classified into region quad-tree, points quad-tree, and edge quad-tree:

- **Region quad-tree.** Region quad-tree represents a partition of space into two dimensions by recursively decomposing the region into four sub-quadrants (equal quadrants). Each node in the tree either has exactly four children, or has no children (a leaf node).

A region quad-tree with a depth of n may be used to represent an image consisting of $2^n \times 2^n$ pixels, where each pixel value is 0 or 1. In addition, region quad-tree may also be used as a variable resolution representation of a data field.

- **Point quad-tree.** Point quad-tree is an adaptation of a binary tree used to represent two dimensional point data. Each node in a point quad-tree represents a real point. The tree shape depends on the order data that is processed.
- **Edge quad-tree.** Edge quad-tree is specifically used to store lines rather than points. Curves are approximated by subdividing cells into a very fine resolution, which can result in extremely unbalanced trees which may defeat the purpose of indexing.

As illustrated in Figure 2.2, the whole space is partitioned into four parts by blue solid lines. Each part is further partitioned into four parts by red dashed lines and so on. The partition goes recursively until each point inside the space can be finally located in a single region.

Quad-tree is suitable for grid-based deployment in sensor networks. In loosing condition, quad-tree can also be applied on uniformly deployed sensor networks. Compared with R-tree, quad-tree has better control on index height and the MBR is

Figure 2.2: Quad-tree

independent of sensor location. To our best knowledge, all the spatial indices working in sensor networks adopt either R-tree or Quad-tree like approaches. The rest of this chapter will give a survey on these major spatial indices in sensor networks.

2.2.1 Peer-Tree

A peer-tree structure [14], loosely speaking, constructs a hierarchical partition of sensor network. A sensor network is partitioned into rectangle-shaped clusters based on the number of nodes contained in a cluster. It uses a bottom-up phase to generate the hierarchical R-tree like structure. Every node cooperates at level 0 of the peer-tree construction. Nodes which are close to each other are grouped together and one of them is selected as head node which acts as the parent of that group. Then, the head node tries to find the nearest head node and the previous step is repeated. Only the cluster heads of level i cooperate to construct the level $i + 1$ of a peer-tree. When a head is finally elected to represent all the nodes, the peer-tree is built up.

In a peer-tree, every node s maintains a variable $l(s)$ to denote the highest level of the peer-tree involved. For each level i , where $0 \leq i \leq l(s)$, s maintains a variable $p(s, i)$ to denote the immediate cluster head of node s at level i . In addition, if a node s is the cluster head of level i , it maintains a variable $c(s, i)$ to denote its children.

Peer-tree also considers self-stabilizing in terms of two actions: *join/form* a cluster and *split* a cluster which are introduced by the author. When a node s realizes that it does not belong to any cluster, it invokes *join/form* cluster action. This node first searches its neighbors at level $l(s) + 1$ by searching an increasingly larger radius. If any node v at level $l(s) + 1$ is found, node s joins v 's cluster. Otherwise, node s becomes a cluster head and sets $l(s) = l(s) + 1$. In cluster *split* action, if a cluster head s at level i has more than N children ($|c(s, i)| > N$), it splits its cluster into clusters with less children. After the splitting, new cluster heads at level i inform their parents about the splitting as the splitting at level i may lead to a further splitting at level $i + 1$.

In order to achieve minimal energy consumption, peer-tree allows cluster heads at each level to send the query to their children parallelly and select the reply with the minimum distance as the result of the nearest neighbor query. The cluster head at each level sends query to its children in a sequential order and only one of the children executes the query at a given time. It is clear that there is no way to obtain both optimizations on response time and energy consumption. A tradeoff must be made.

The design of peer-tree is concerned with efficient execution of nearest neighbor (NN) queries in sensor network. Particularly, peer-tree structure allows the query to be presented at any node instead of a sink node only. In query execution, depending on the tradeoff, the query is executed in different ways. The author does not specify how the query results are sent back to the query generating node. We assume the

Figure 2.3: SPIX tree

query sending and data collecting adopt the same path. In minimal response time mode, cluster heads may fail earlier than their child nodes that may lead to child nodes being disconnected disconnect from the network. Although a join action is provided, the cost is not trivial.

2.2.2 Spatial Index Tree

The ability to run spatial queries is extremely useful for sensor networks. However, processing spatial queries in sensor networks differs from a spatial database. In SPatial IndeX (SPIX) [52], the authors adopt a R-tree like structure to index sensor networks. The sink node acts as the root of the tree and each parent keeps its children's MBR information as the R-tree does. All queries are injected from the sink node. The sink node is responsible for spreading the query in a sensor network and getting the result back.

Each sensor node in SPIX maintains a Minimum Bounded Area (MBA) which covers its offspring and itself. When a node hears a spatial query, it compares the query area with its MBA. If the intersection is not empty, query is forwarded to its children. Otherwise, it drops the query. The authors point out that sensors with smaller MBA have a higher chance to determine whether a spatial query applies to them or not.

SPIX exploits Rectangle Model and Angular Model to create a routing tree. In the rectangle model, MBA is the MBR that covers the node and all the nodes below it. In the angular model, MBA is the minimum bounded pie represented by start/end radius and start/end angles. When the spatial query is based on the distance between sink node and sensors, angular model is more efficient than rectangle model.

Considering that sensor nodes are distributed in the network and do not have a global view of the whole network, SPIX illustrates ways to systematically construct a spatial index in sensor network which is maintained by sensors to efficiently evaluate spatial queries. The building process of SPIX-tree can be divided into two steps: *Advertisement* phase and *Parent selection* phase.

In the *advertisement* phase, sink node broadcasts itself to the network. Each node in the network will not broadcast itself to the network until it receives an advertisement. This phase will stop when all the nodes have received an advertisement. In *parent selection* phase, the nodes without any child candidate pick up a parent first. A parent node will not pick up its parent until all its children have finished selecting parents. Particularly, in the *parent selection* phase, a sensor always selects a parent whose MBA needs the least area enlargement to include itself. If there are more than one parent candidates with the same MBA enlargement, it chooses the nearest one. After the two construction phases, each parent knows its offspring and has an overview of their MBA, as shown in Figure 2.3. Each parent in SPIX-tree

Figure 2.4: Effect of eliminating thin rectangles

maintains the total MBA (illustrated by rectangle) of its offspring. When a spatial query arrives, a parent can easily decide whether to forward this query to its children or not.

Considering there might be undesirable long range radio communication, thin rectangles need to be removed after the two phase construction. When a child node finds that it is closer to its grandparent than its parent, it disconnects from its parent and selects its grandparent as the new parent as illustrated in Figure 2.4:

As the authors point out that, since a smaller MBA has a higher chance of determining whether a spatial query applies to it or not, the energy optimization phase is provided to split large MBAs into small MBAs. When a leaf node determines that it is located in another node's MBA area, as illustrated in Figure 2.5, it runs "parent-switching verification" process. In this process, the sensor network moves some nodes from one branch to another to reduce the MBA of the parent node. Since the smaller the MBA is, the less chance it will be involved in a spatial query, energy efficient is achieved in this way.

Figure 2.5: Effect of energy optimization phase

Node which needs to join the network after the SPIX is built up runs the join phase. By broadcasting the join request, a node can get a list of parent candidates. The node chooses its parent that results in the minimal MBA enlargement, which is exactly the same with parent selection phase.

SPIX guarantees the query spreading involves less non-relevant nodes, especially after the optimization phase. The join phase can further prolong the network life time to some extent. However, new node join will cause all its ancestors to update their maintained MBA information. It may cause flooding which is not trivial in the sensor network.

2.2.3 OMSI-Tree

OMSI tree [63] is quite similar with SPIX [52]. They both adopt a R-tree like structure and use two similar phases in their tree construction process (*topdown* and *bottomup* phases in OMSI-tree).

The difference is that SPIX considers tree structure optimization by eliminating thin rectangles and cutting off leaf nodes to avoid large MBR in construction phase, while OMSI-tree focuses on recording the nodes which overlap with one another. Parent node records the MBR of itself as well as its children's total MBRs so that it can have enough information to determine if all its children need to be involved in a given spatial query.

Figure 2.6: Example of OMSI-tree

OMSI-tree considers two overlapping situations, namely, overlap on the vertices of query window and overlap on the borders of the query window. For the first situation, if the vertex falls into the overlapping region, relevant sensor nodes should be taken into consideration. Suppose that sensor node A has MBR_A , sensor node B has MBR_B and A, B have overlapping area MBR_{AB} , and the query window is W . The processing steps are: first, compare W with MBR_A . If the overlapping area MBR_{WA} belongs to MBR_{AB} completely, it is convinced that node A cannot provide any available information in this query. Thus, sensor node A is abandoned and B is selected. Otherwise, it keeps sensor A and checks sensor B in the same way. There is no specified order for checking sensors A and B .

In another situation, when the overlapping area appears on the borders of the query window, only the sensor node which is physically inside the query window will be kept. It means this is decided by their geographical positions.

A spatial query based on OMSI-tree is illustrated in Figure 2.6. An OMSI-Tree has already been built up and a spatial query is given. Obviously, there are three

vertices placed on the overlapping area. For the first one, sensor node A and B share the overlapping area MBR_{AB} . It is clear that the overlapping space between sensor node A and query window W completely belongs to MBR_{AB} . According to their overlapping MBR removing (OMR) algorithm, sensor node A is abandoned. Sensor node K is abandoned in the same way. While more range can be detected in query window W except MBR_{DM} for sensor node D . Based on the OMR algorithm, sensor node D is kept and the relevant sensor node M is checked. Evidently, sensor node M is also kept.

The OMSI-tree [63] brings the capability of executing spatial query in sensor networks and reduces the number of involved sensor nodes when there are redundant nodes in the network. However, OMSI-tree neither considers any tree structure optimization nor node rejoin actions. Besides, OMSI-tree adopts the same path for query spreading and data transmitting. That is why the performance decrease suddenly when certain proportions of intermediate nodes fail as shown in the experiments.

2.2.4 Subarea Tree Routing

Subarea Tree Routing (STR) [37] is also a kind of hierarchical spatial index. It first identifies root nodes of the network. Then, root nodes start the process of establishing subarea trees. Every node either belongs to a subarea tree or becomes an interconnect node after the construction phase.

There are two approaches for root node identification, namely, static method and auto-discovery. Static method adopts manual configuration which requires pre-knowledge of the whole network. Auto-discovery process is more complex. The requirements of becoming a root node must be given first. Then, from an arbitrary node, the requirements are broadcast into the entire network. Any node that meets the requirements will become a root node and broadcast itself into the network. Thus, every root node knows the other root nodes.

The process of establishing subarea trees is through a sequence of message exchange. The following two messages are hired for tree construction: Tree Establishing Message (STEM) and Tree Node Updating Message (TNUM). In the beginning, every node periodically broadcasts *STEM* which includes node ID, node type, the number of offspring and so on. A node that receives a *STEM* will set the sender as parent and return a parent *TNUM* which includes necessary information for a parent to update its children list. If a node cannot join a subarea tree after this process, it turns itself an interconnected node.

The STR construction process is shown in Figure 2.7. Figure 2.7 (a) shows a network of 22 nodes with unique ID. The lines denote the wireless link between two nodes. Given the metric that a node becomes a root node if the number of its neighbors is more than or equal to 4, node *S* broadcasts Root Auto-discovering request (RADRQ) message and nodes *A*, *B*, *C* will become root nodes. These three nodes will know each other by broadcasting their information.

By referring to the existing information, routing information is built up as shown in Figure 2.7 (b). $A \leftrightarrow B$, $A \leftrightarrow D \leftrightarrow C$ and $B \leftrightarrow E \leftrightarrow F \leftrightarrow C$ are the routing paths among the root nodes. Nodes *D*, *E*, *F* become the interconnect nodes.

Root nodes *A*, *B*, *C* start the subarea tree construction by broadcasting *STEM*. The rest of the nodes who receive this message will join the subarea tree and send out a *STEM* message as well. As illustrated in Figure 2.7 (c), all the nodes belong to a subarea tree except nodes *H*, *I* and *R*. Considering this situation, STR turns an interconnect node into a root node if nodes cannot join a subarea tree after a certain time threshold. In this case, node *E* is turned into a root node and will gain other root nodes' information through the nearest root node *B*. The rest of the root nodes will also be informed that node *B* has been upgraded to root node.

A data delivery example is shown in Figure 2.7 (d). Assume a data packet needs to be sent to node *R* from node *S*. The data packet first arrives in its parent node

Figure 2.7: Construction of STR

G . G will send this packet to root node A once it has found that node R is not one of its offspring. Root node A will send Routing Inquiry Message (RIM) to root node B . B will forward RIM to E if it fails to find the routing information for node R . Root node E will respond to root node A with a Routing Reply Message (RRM) because node R is its offspring. Once it has received the RRM message from root node E , Root node A will send the data packet to root node E . The data packet is finally delivered to node R through node H .

Figure 2.8: Naive routing tree versus Sectioned tree

A well organized subarea tree is able to deliver a message to the destination node through the shortest path to achieve a better performance. However, this greedy algorithm will cause the interconnected nodes or root node to fail early. The maintenance of subarea trees is not trivial due to their message exchange process. Since the network scale tested in the experiment was really small, we cannot determine the performance in a large scale network.

2.2.5 Sectioned Tree

Sectioned tree [47] divides the network into virtual sections. Each section maintains a local tree to prevent several branches from forwarding queries and data which may consume extra energy.

As illustrated in Figure 2.8 (a), a query for area *A* is injected into the network from the sink node. Most of the nodes (including *s1*) are able to receive the query from the node *sr1*. However, node *s2* is not able to receive the query from *sr1* even if it is close to node *s1*. Node *s2* has to receive the query from node *sr2*. All the

irrelevant nodes (white node) have to be involved to pass the query to node $s2$. These involved irrelevant nodes could conserve their limited energy if node $s1$ could pass the query to $s2$ directly.

Motivated by this, the authors proposed their idea to divide the network into virtual sections and each virtual section maintains a local tree. The advantages of doing this are: virtual sections prevent several branches from forwarding queries or data that might consume extra energy; each section maintains a local tree which is scalable since it only cares about the limited nodes within the same virtual section.

The construction of local trees contains three steps. Firstly, each node calculates the distance to the anchor point. The closest point will be elected as the root of local tree. Secondly, each local tree root starts building a local temporary tree by broadcasting its information to the rest of nodes. Generally speaking, the local temporary tree adopts two steps which are similar to the OMSI-tree and SPIX construction process. Thirdly, either Dijkstra's shortest path algorithm or Prim's MST is adopted to build energy efficient local trees. The shapes of the trees constructed by these two algorithms differ from one another. The initial radio range of nodes can be dominant to determine the shape of the tree when using Dijkstra's shortest path algorithm. While for the tree constructed by Prim's MST, the initial range of sensor nodes does not affect the shape of the tree since it always looks for the closest node which is not in the spanning tree.

A sectioned tree is illustrated in Figure 2.8(b). Node $s1$ and node $s2$ are located in the same virtual section and connected by the local tree. Node $s2$ can receive the query from $s1$ through local tree instead of receiving through node $sr2$. All the energy spending on the white node is conserved and energy efficiency is achieved in this way.

The construction of a sectioned tree contains three phases. In phase I, global information is exchanged globally and local information is exchanged inside each

Figure 2.9: Distributed quad-tree

section. In phase II, local trees are constructed either by Dijkstra’s shortest path algorithm or Prim’s MST algorithm. In phase III, local trees are connected to form a global tree. MBR for spatial indexing is built up in the last phase.

Sectioned tree considers connecting the neighboring nodes in terms of local trees to prevent several branches from forwarding queries and data. To some extent, sectioned tree is able to conserve energy of the entire network. However, sectioned tree is not able to prevent unnecessary energy consumption completely. Nodes on the boundary of the sections belong to different local trees. Queries to these nodes have to go through more than one local trees which is still not energy efficient.

2.2.6 Distributed Quad-Tree

With the assumption that sensor nodes are distributed in a grid space, Distributed Quad-tree (DQT) [39] structure limits the cost of answering a query to a constant factor $2\sqrt{2}$ of the distance “ d ” to the nearest event in the network. In addition, the authors claimed their DQT is deployable in practice by satisfying the following three requirements: distance-sensitive for querying and efficient for information storage; low construction cost; ability of facing node failures.

DQT introduces the concept of level 1 box which is the smallest cell in DQT and all the nodes inside it are within one hop distance. Every four level 1 box nodes are clustered together to form a level 2 box node. One of the four level 1 box nodes is chosen to be the cluster-head of that level 2 box node. In other words, a node is elected as cluster-head in each level and the cluster-head is always its own child in a lower level. The node which is closest to the geographic center of the network is always elected as cluster-head at each level as illustrated in Figure 2.9. Node 003 is elected as cluster-head for 00 region at level 1 because it is closest to the center. Similarly, node 033 is elected as cluster-head of level 2 because it is closer to the center than level 2 nodes 003, 013 and 023. Nodes 122, 211 and 300 are elected as level 2 cluster-head in the same way. Backward link can be avoided by electing cluster-heads in this way.

In order to map sensor nodes from physical position to logical nodes in DQT, the authors assume that each sensor node is aware of its physical coordinates (x, y) . In addition, each cell in DQT is represented by two endpoints (x_s, y_s) and (x_e, y_e) . Assume that DQT has i levels and the area of each level 1 box is $w \times l$, where width $w = (y_e - y_s)/2^i$ and length $l = (x_e - x_s)/2^i$, a node (x, y) can be mapped to DQT by following formula:

$$DQT_{addr} = [\lfloor \frac{x - x_s}{w} \rfloor (mod 2)] + [\lfloor \frac{y - y_s}{l} \rfloor (mod 2)] * 2$$

Instead of bottom-up construction, DQT uses local construction to avoid expensive communication when building up the spatial index. Each node records its 8 nearby neighbors at eight directions: N, S, E, W, NE, NW, SE, SW. Their cluster-head validate algorithm uses this information to select a cluster-head and guarantees the cluster-head at each level is closer to the map center than any of its children.

Considering some nodes may be physically close to each other but logically far away from each other, the idea of *sibling link* is introduced. The sibling link is the link between a node and its neighbors in each direction and only exists between nodes at the same level. A packet from node 011 to 100 can go through path $011 \leftrightarrow 013 \leftrightarrow 102 \leftrightarrow 100$ instead of path $011 \leftrightarrow 013 \leftrightarrow 033 \leftrightarrow 122 \leftrightarrow 102 \leftrightarrow 100$.

DQT uses the same structure as a Quad-tree, namely, each parent has exactly 4 children and each child has another 4 children. Thus, DQT could maintain itself in a low height while indexing many sensor nodes. This characteristic is quite helpful to reduce communication in the network. However, in the real world, sensor networks are more likely to be deployed randomly. Besides, the authors assume that the sink node is always at the center of the network which is quite unlikely to be true in practice. Thus, there are not many situations to which DQT can be applied. Moreover, similar with other approaches, the query forwarding path is the same with the data transmitting path.

2.2.7 Quadtree-based Data Dissemination

Quadtree-based Data Dissemination (QDD) [46] is a quadtree-based network space partition, which is motivated by two goals. One is to accommodate the dynamicity and to avoid frequently updating the index due to stimulus and sink mobility. The other one is to keep the index simple.

By assuming that every node is aware of its location and total sensor network space, a sensor node recursively divides the network into four parts until it is the only node of that sub-partition. The partition process is shown in Figure 2.10. Sensor node S takes the whole sensor network space N as the root and logically partitions it into four equal sized quadrants, North West (NW), South West (SW), North East (NE) and South East (SE). Each of them corresponds to a child of N . Each sub-quadrant is considered as a separate parent and is further divided into four child

Figure 2.10: Partition of QDD

quadrants. This process is repeated until node S is the only node in a sub-quadrant. Finally, each leaf quadrant corresponds to a sensor node.

In data forwarding phase, whenever a node s detects a stimulus, it performs a logical partition of the sensor network. Each partition at level i is represented by endpoints denoted as $(i.X_{LB}, i.Y_{LB})$ and $(i.X_{UB}, i.Y_{UB})$. In addition, the corresponding Rendezvous Point (also called central point) of each partition is given as (x_i, y_i) where

$$x_i = i.X_{LB} + (i.X_{UB} - i.X_{LB})/2$$

$$y_i = i.Y_{LB} + (i.Y_{UB} - i.Y_{LB})/2$$

Node s first sends data message to its immediate rendezvous point and the immediate rendezvous point further forwards the message to the rendezvous point at a higher level. The process is repeated until the data message arrives at the desti-

nation. During this process, every rendezvous point records the message that it has forwarded for a certain time to prevent duplicate entries or re-query.

In the query forwarding phase, a sink node sends the query to its immediate rendezvous point. On receiving this query, immediate rendezvous point checks whether it has the data. If so, the immediate rendezvous point returns the data to the sink node. Otherwise, immediate rendezvous point forwards the query to the rendezvous point at a higher level. The rendezvous point at the higher level will repeat this process until it finds the data.

Generally speaking, data forwarding and query forwarding steps are similar. Whenever an event happens, a node first contacts its immediate rendezvous point. If the immediate rendezvous point cannot finish the job, the rendezvous point at a higher level will be involved.

By assuming that each node is aware of its location and the total sensor network space, sensor nodes can elect the node that is closest to the center to be subtree parent. Thus, QDD does not require sensor nodes to be in a grid-based manner. This advantage brings more situations to which a quad-tree like spatial index can be applied. However, it still faces the problem of parent node failure. Besides, in order to work on an arbitrary node deployment, QDD calculates the partition of network every time when there is data or query forwarding action, which is not energy efficient.

2.2.8 Conclusion

As discussed above, we have noted that almost all the spatial indices adopt a hierarchical structure to keep nodes in a parent-child relationship. The most frequently used are R-tree [14, 52, 63, 37, 47] and Quad-tree [39, 46] like structures. Both structures are common in that each parent node indexes its offspring in terms of

Figure 2.11: Greedy routing

MBR information. By referring the index, query could be prevented from flooding all over the sensor network and only be sent to the relevant nodes. This is quite energy efficient, especially when the query window is small.

Generally, quad-tree like structures have better performance than R-tree like structures when sensor nodes are deployed in a grid space or uniformly deployed. While in practice, it is unlikely to know and guarantee that sensor nodes are deployed in a grid space or are uniformly deployed. This explains why most works adopt R-tree like structures to build up spatial index in sensor networks. All these works have shown that by building up a spatial index, sensor networks are able to support spatial query which is more energy efficient than the traditional approach.

However, few of them have noted that the parent node failure may lead to offspring nodes to fail even if they are still in good health. Although some of them have provided node rejoin action, they have ignored the fact that the cost of updating spatial index is not trivial. In next chapter, we will introduce our multi-parents method to keep the parent node alive as long as possible, which in turn, prolongs the lifespan of the sensor network.

Figure 2.12: Greedy routing failure

2.3 Geographic Routing Based Approach

Geographic routing is also known as stateless Routing. In general, a geographic routing algorithm consists of two stages: greedy routing and face routing.

2.3.1 Greedy Routing

In greedy routing, each node keeps forwarding packet to one of its neighbors, which is closest to the final destination to ensure that the packet approaches the destination after sufficient forwardings. Figure 2.11 shows a packet arrives at the destination by greedy routing, where dashed circles indicate communication radius of sensor node.

2.3.2 Face Routing

In face routing, a packet usually traverses a series of faces. In each face, if the packet meets a node, which has an edge that intersects with the “virtual line” from the source to the destination, it stops traversing the current face and switches to the next one. Such an operation is referred to as face changing and it is repeated until the packets reach the destination or get back to the greedy routing state. For all the existing geographic routing protocols, they use the same greedy routing method

Figure 2.13: Right hand rule

but differ in face routing phase. In the rest of this thesis, we focus on face routing whenever we mention geographic routing. Face Routing employs the right hand rule running on a planarized graph.

2.3.2.1 Right Hand Rule

Face Routing employs the well-known right hand rule to forward packet on a planarized graph. According to the rule, when a node x receives packet from node y , the outgoing link is the next one sequentially clockwise around x from link yx as illustrated in Figure 2.13. By right hand rule, the packet keeps traversing the interior of a polygonal region (a face) to approach the destination.

Unfortunately, on graphs with cross links, right hand rule may not be able to route packet to the destination or somewhere else where it can resume greedy routing. Due to the characteristics of wireless communication, sensor radio graph is always full of cross links. The sensor radio network must be planarized before applying right hand

rule.

2.3.2.2 Planarization

Most of the existing works assume each sensor node has the same communication radius. With such an assumption, planarizing the network is a process of removing selected links from the sensor radio network. GG (Gabriel Graph) and RNG (Relative Neighborhood Graph) are most commonly used because they can be executed in a distributed manner.

The GG is defined as follows:

A link uv belongs to GG if there are no other nodes located inside the circle whose diameter is uv , as illustrated in Figure 2.14.a. Given a sensor network with sensor set V where every node maintains a neighbor list N , each sensor node u can remove non-GG links by running Algorithm 1.

Algorithm 1 Planarize by GG

```

1: for  $v \in N$  do
2:    $m \leftarrow$  middle point of link  $uv$ 
3:   for  $w \in V$  do
4:     if  $w \neq u$  and  $w \neq v$  and  $|mw| < |mu|$  then
5:       remove link  $uv$ 
6:       break
7:     end if
8:   end for
9: end for

```

The RNG is defined as follows:

A link uv belongs to RNG if there are no other nodes located inside the overlap communication area of node u and v , as illustrated by Figure 2.14.b. Given a sensor network with sensor set V where each node maintains a neighbor list N , each sensor node u can remove non-RNG links by running Algorithm 2.

Both GG and RNG can be used to construct a planar graph. However, applying on the same radio network, GG always extracts a relatively fine grained subgraph.

2.14.a: Gabriel graph

2.14.b: Relative neighborhood graph

Figure 2.14: Gabriel graph and relative neighborhood graph

Algorithm 2 Planarize by RNG

```

1: for  $v \in N$  do
2:   for  $w \in V$  do
3:     if  $w \neq u$  and  $w \neq v$  and  $|uv| > \max[|wu|, |vu|]$  then
4:       remove link  $uv$ 
5:       break
6:     end if
7:   end for
8: end for

```

Therefore, most of the existing works choose GG rather than RNG. We also use GG instead of RNG in the remainder of this work.

2.3.2.3 Face Routing Variants

Many face routing variants, which differ in how to make face change decisions, have been implemented. Generally, face routing variants can be classified into two categories, those that make face change decisions as soon as the condition is met and those that require fully exploring the face before making a face change decision.

Fully exploring the face gives the optimal decision. An example can be referred to Figure 2.15. Imaginary line st intersects face F_1 at points p_1 , p_2 and p_3 . Face routing variants which make the face change decision as soon as they meet changing conditions will make face change decision at p_1 to start exploring face F_2 and back to face F_1 again at intersection p_2 . While face routing variants that make fully face exploration will make face change decision at intersection p_3 . Apparently, fully exploring the face before making a face change decision is more efficient in this case because it avoids traversing face F_2 . However, it may not be efficient if most of the faces are small faces with two intersections only, such as face F_3 .

Face routing which partially explores face results in less overheads but can not guarantee delivery alone. Figure 2.15 illustrates an example of face routing failure. Node B detects link BE intersects line st (blue cross) and makes face change decision. Previously, packet was traversing on external face. When the face change decision is made, the packet starts to traverse interior face F_{BDE} . While traversing face F_{BDE} , another intersection (green cross) is detected. However, it is further to the destination than where the packet entered face F_{BDE} , so a face change decision can not be made. Thus, packet keeps looping inside face F_{BDE} because an intersection which is closer to destination than where it enters this face can not be found.

Figure 2.15: Face routing failure

Figure 2.16: Face routing part of existing geographic routing algorithms

Geographic routing algorithms that make stateless face change decision:

GPSR [28] simply explores faces that have an intersections with the virtual line st (source and destination pair) using the right hand rule (left hand rule is applicable as well). When face traversing meets a node with an edge intersecting with line st , which is closer to the destination than the previous intersection, GPSR starts to explore the next face. As there are limited edges in the network, packets are guaranteed to reach the destination.

Refer the network topology in Figure 2.16 for an example. Suppose a packet is sent from s to t . When the packet traverses face F_1 using right hand rule, the first node intersecting with line st is v_2 . Face exploration changes to face F_2 at v_2 . While traversing F_2 , packet meets v_{11} and changes face exploration back to face F_1 . When exploring F_1 from v_{11} , the packet encounters v_{12} and the face exploration changes to F_3 . At F_3 , the packet changes to F_4 from v_{18} . At F_4 , the packet is able to reach the final destination t . In total, the sequence of visited faces is F_1, F_2, F_1, F_3 and F_4 .

Greedy-Face-Greedy(GFG) [6] is slightly different from GPSR. They produce similar paths most of the time. Whenever a packet encounters an edge intersecting with lines st at point p , it will switch to the immediate face which intersects with the open line segment pt . Point p_1, p_2, p_3 and p_4 in Figure 2.16 are the intersections. The sequence of visited faces is the same as GPSR in this example.

Greedy Path Vector Face Routing (GPVFR) [34] does not use a fixed line st for all the face intersections check. Like GFG, GPVFR changes to the next face as long as the packet meets a node with edge intersecting with st . The node with intersection will be treated as a new start node to serve as a new virtual line. As illustrated in Figure 2.16, a packet is sent from s to destination t . When the packet arrives at v_2 , it encounters the first intersection p_1 and the face exploration changes to F_2 . Starting from F_2 , v_2 is treated as the starting point and line v_2t is the new guide line. As the

same procedure repeats, intersections q_1 , q_2 and q_3 are encountered. Eventually, the packet changes to F_4 from v_{19} and reaches the destination.

Geographic routing algorithms that require complete face routing before making face change decision:

Compass Routing [31] requires a packet to traverse the entire face before making a face change decision. There could be more than one edge that intersects with line st . These intersections are compared and the packet is advanced to the edge with the closest intersection to destination. The face which shares that edge with the current face is selected as the next face to explore.

As the example in Figure 2.16, F_1 will be fully explored by a packet sent from s to t . When the packet is back to s , the packet will be advanced to the edge that p_3 is on, and face exploration changes to F_3 . When exploring F_3 , p_4 is chosen as the closest interaction and F_4 is next explored. Eventually, the packet arrives at t from F_4 . The sequence of visited faces is F_1 , F_3 and F_4 .

Greedy Other Adaptive Face Routing (GOAFR+) [33] requires complete face exploration as well. Packets are advanced to the node which is the closest to the destination on the face and face traversal is restarted. In the example of Figure 2.16, face exploration changes from F_1 to F_4 at v_{20} since it is the closest node to the destination on F_1 . The visited face sequence is F_1 and F_4 .

Chapter 3

Multi-parents Approach

In existing work so far, almost all spatial indices adopt a hierarchical structure (e.g., R-tree [5], Quad-tree [2]) with the sinknode as the root of the tree. Every non-leaf node maintains its subtree MBR and children information. Query evaluation starts from the root and propagates to parts of the tree. When a query reaches a node, the node compares the query's spatial coverage with its MBR. If the spatial coverage intersects with its MBR, the node forwards the query to its child nodes and so on. If there is no overlap, the node abandons the query.

It has been shown in existing work that the use of spatial indices to support queries indeed prolong the lifespan of sensor network by excluding irrelevant sensor nodes during query evaluation. However, to the best of our knowledge, existing work uses the same node path for propagating query and routing back sensor data. Hence, intermediate nodes that have many child nodes incur more communication operations than non-intermediate nodes. When the battery life of intermediate nodes are drained, the entire subtree subtending from these nodes is disconnected from the rest of the sensor network. If no rejoin methods are provided, the disconnected subtree can never be visited anymore even if there are still active nodes in the subtree. Furthermore, even if rejoin methods are provided, the cost of updating the whole spatial index is not trivial.

If there are alternative return paths back to the sink nodes, we believe the lifespan of intermediate nodes can be prolonged because the workload of intermediate nodes is reduced; thus, extending the lifespan of the entire sensor network. In this chapter, we propose to prolong the lifespan of a sensor network by allowing sensor nodes to have more than one parent so that there are alternative return paths for sensor data to be routed back to sink nodes. Various metrics are provided for nodes to choose which specific parent node to use.

3.1 Multi-parent Methods

In this chapter, we describe our proposed multi-parent approach to reduce the workload of intermediate nodes (parent nodes). We also describe three spatial indices that support spatial queries. They are the OMSI-tree, nearest parent tree and the hybrid tree. We will start with the three spatial indices first as their structures give a better picture of the motivation of our proposed approach.

3.1.1 OMSI-Tree

This index has two phases of construction: descending and ascending phases, as shown in Figure 3.1; solid lines represent the descending phase, and dashed lines represent the ascending phase. The rectangles are the MBRs of sensor nodes. The parent data structure has the following information $\langle \text{ParentPts}, \text{ChildPts}, \text{ChildMBRs}, \text{OverallMBR}, \text{Location-info} \rangle$.

In the descending phase, the message, which includes sender location and ID, is disseminated from the sinknode. All sensor nodes will not broadcast themselves to the sub-net within its radio range until they receive a message. All nodes keep a list of messages it has received. The descending phase completes when all sensor nodes receive a message.

Figure 3.1: OMSI-tree

In the ascending phase, when a sensor node has no children candidate nodes, it chooses its parent node; otherwise, it waits until all the children candidates select their parent nodes. We refer to a node that has no children candidate nodes as a leaf node. When a leaf node confirms its parent node, it updates its MBR to the parent node. When a parent node has collected all the children MBRs under itself, it computes the overall MBR and select its parent node. Thus, a distributed R-tree like structure is constructed among the sensor nodes.

3.1.2 Nearest-Parent Tree

Compared to the OMSI-tree, the construction of this index is relatively easy. Sinknode broadcasts its location information to the network. Once a node receives the broadcast call from the sinknode, it detects the nodes around its neighborhood and keeps a reachable node list. After a certain time period, sensor node compares the candidates in the list and picks the nearest neighbor that is closer to the sinknode than itself as its parent. If all the nodes that are reachable by this node are further away than this node from the sinknode, it will weaken the selection rule to pick the

Figure 3.2: Nearest parent tree

nearest neighbor as the parent as each node always tries to find the nearest neighbor as the parent. There is the possibility that nodes that are directly reachable to the sinknode pick other nodes as its parent. As sinknode has no battery-life problem, the disadvantage is obvious.

The topology of nearest parent tree is shown in Figure 3.2. Comparing with OMSI-tree, each parent has less children. This reduces parent workload to some extent. However, it also fails in leveraging on the advantage of sinknode power ascendancy.

Figure 3.3: Hybrid tree

3.1.3 Hybrid Structure

Hybrid structure takes the advantages from both OMSI-tree and nearest parent tree. This index improves the parent selection metric. All nodes that are directly reachable to the sinknode pick the sinknode as its parent. This is to leverage on the advantage of sinknode power ascendancy. Those nodes which are not directly reachable to sinknode selects the nearest parent in order to reduce the workload of parent.

Figure 3.3 illustrates the topology of hybrid tree building on the same set of sensor nodes with nearest parent tree (100 nodes in 20×20 space). It is obvious that more nodes are directly connected to the sinknode in the hybrid tree. This characteristic is very useful in our proposed multi-parent methods (shown in experimental results

part).

From Figures 3.1, 3.2 and 3.3, we observe that the structure of the OMSI-tree is quite different from the other two. Each parent maintains more children than the other structures. These three spatial indices will serve as the platform which we evaluate the multi-parent methods in the rest of chapter.

3.1.4 Multi-Parent Methods

For a given sensor network, it makes sense to have a flatter hierarchical spatial index for the nodes as this means that each parent node has more child nodes, and consequently, less number of hops is required to deliver a query message in the query forwarding phase.

However, this also means that parent nodes have heavier workloads and shorter lifespan. Considering that parent maintains its offspring's information, failure of a parent node means that losing information of the offspring. Queries cannot be forwarded to a dead branch unless spatial index is re-built, which is not trivial.

To further extend the lifespan of a sensor network, we propose to take the return path of query-result into consideration. The basic idea is that each child node should keep a list of parent candidates. When sending result data back to sinknode, a child node selects one of the parent candidates to send back the data. The selected parent, which is also a child node of other parents, also has multiple choices of parent nodes. In this way, multiple nodes share part of the workload. We refer to this as the multi-parent approach.

As there is now a choice of parent nodes, there are different ways to pick the parent node. We propose four ways: (1) pick parent nodes in a round-robin manner (like a token ring); (2) pick the parent that is physically nearer; (3) pick the parent that has the most remaining energy; and (4) a combination of (2) and (3).

(a)

(b)

(c)

Figure 3.4: Experiment results of (a) OMSI-tree, (b) Nearest parent tree and (c) Hybrid tree with radio radius = 2

(a)

(b)

(c)

Figure 3.5: Experiment results of (a) OMSI-tree, (b) Nearest parent tree and (c) Hybrid tree with radio radius = 4

(a)

(b)

(c)

Figure 3.6: Experiment results of (a) OMSI-tree, (b) Nearest parent tree and (c) Hybrid tree with radio radius = 6

The motivation of selecting the nearest parent is the fact that the longer distance a message transmits, the more energy it incurs. The energy cost of sending one message over a distance of d is $e = \kappa d^c$ where κ and c are constants and c is usually between 2 and 4. It is clear that multi-hops transmission is more energy efficient than one-hop transmission and the less distance per hop, the less energy is incurred.

Even with the multi-parent approach, child nodes still cannot be visited if their parent node died even though they are in good health status. That is why we propose another selection strategy: Pick the parent who has the most remaining energy. Having more parents alive means that more child nodes can be visited.

Combining the above two strategies, we propose another strategy: Pick the parent with the highest ratio E/d where E is the total remaining energy that a parent candidate node has and d is the distance to that parent candidate. This leverages on both advantages when finding a parent.

Besides considering distance and energy factors, contacting each parent candidate uniformly also helps to improve the parent's lifespan. Our approach is to maintain a token ring on parent candidates so that each parent candidate has equal chance to be selected.

All the four strategies focused on sharing the workload of parents in the query data (result) transmitting phase. The additional energy cost, comparing with the conventional single parent approach, is zero except for the energy used to maintain parent candidates list at the child node, which is low and negligible.

Using the multi-parent approach, querying a sensor network goes like this: The query is injected to the sensor network from the sinknode. Each parent near the sinknode compares its MBR with the query's spatial coverage. If its MBR intersects with the query's spatial coverage, it forwards the query to its child nodes. Once all the sensor nodes which fall within the query's spatial coverage are read, they begin

to transmit the query result back to their parents. In the multi-parent approach, each node now pick a parent node to carry the result data back to the sinknode. In next section, we will empirically confirm that the multi-parent strategies prolong the lifespan of sensor networks.

3.2 Experimental Results

In this section, we evaluate the performance of the proposed multi-parent methods on sensor networks with respect to different density and scalability of nodes. We use a Blade 32-node cluster, each CPU is a Dual Core Xeon 3.0GHz, 4GB RAM for each node, running the Rock 4.2.1 task management system. The programs are coded in Java. We use the MicaZ node specification (2, 000mAh battery, sending and receiving consume 140.91mW energy and 36mW when radio is off) for the sensor nodes. The sensor nodes are deployed in a 20×20 grid; the number of sensor nodes ranges from 50 to 350.

Sensor Network Failure

We determine that a sensor network is no longer accessible, when 25% of all sensor nodes are dead or 1 million queries have been executed. At the node level, when a sensor node exhausts all of its energy, it is considered dead. In single-parent methods, parent failure would mean that all its descendent nodes fail as they can no longer reach the sinknode. In multi-parent methods, a node that fails will not cause other nodes to fail.

Query Set

We synthetically generate a set of 1 million queries with the following distribution of query spatial coverage: 87% of the queries are querying less than 25% of the total

network area; 11% of the queries are querying less than 50% of the total network area; 1.5% of the queries are querying less than 75% of the total network area.

Other Parameters

We vary the **node density** in the 20×20 unit grid from 50 (low) to 350 (high). For each density, the sensor nodes are randomly created in the grid. For all experiments, this is repeated 3 times and averages are taken for all values taken. We also vary the **sensor radio radius**: 2 units(inadequate), 4 units(adequate), 6 units(excessive). Each experiment is performed for the single-parent without aggregation method, single-parent with aggregation, and the 4 multi-parent methods proposed. We have included the single-parent with aggregation as existing work [32], [12] have shown that in-network aggregation can help reduce communication in the network.

Results

The experimental results are shown in Figures 3.4, 3.5 and 3.6. Each figure is a plot of the number of queries that can be executed before the network fails versus the density of nodes in the network. From Figure 3.4, we note that the multi-parent methods perform better than the single-parent without aggregation, but worse than the one with aggregation. The reason is that in this low radius model, each sensor does not have many parent candidates. Hence, the multi-parent methods do not perform well.

When we increase the node radio radius to 4 units (Figure 3.5), it is obvious that all three spatial indices benefit from the multi-parent methods. The single-parent method without aggregation continues to have the worst performance. With in-network aggregation, the single-parent method sometimes performs better than the multi-parent methods. However, we note that the average performance of the

multi-parent methods using the OMSI-tree is much better than the single-parent methods. The network lifespan is almost 2 times longer than when the single parent method is used. For the other two spatial indices, the multi-parent methods showed improvements. For the hybrid structure, the multi-parent methods almost always have the same performance as the single-parent method; the differences between the single-parent with aggregation method are narrowed. Hence, when the sensor radius is adequate (4 units), the multi-parent methods work best to prolong the lifespan of the network.

When we further increase the sensor radio radius to 6 units (Figure 3.6), we can see that such increase further prolongs network lifespan when using the OMSI-tree spatial structure. Multi-parent methods maintained the network lifespan 3 times longer than the single-parent methods. Compared to a radius of 4 units, the multi-parent methods sustained the lifespan of the network for an additional 250,000 queries. The performance of the multi-parent methods surpassed the single parent methods under the hybrid spatial index. When the radius is 4 units, multi-parent methods almost always have the same performance with the single-parent with aggregation method. However, for a radius of 6 units, multi-parent methods always performed better than the single-parent methods; the network lifespan is almost 2 times longer.

From the above results, we note that for all radius values, the multi-parent methods under the nearest parent tree always perform poorly compared to the other two spatial indices. By analyzing the topology of nearest-parent tree indexed sensor networks, we found that the tree cannot be completely constructed when radio coverage is low; the network is disintegrated into several disconnected portions. Also, very few nodes are connecting to the sinknode. Hence, the energy advantage of the sinknode is lost. Our conclusion is that the multi-parent methods are not feasible for networks relying on the nearest parent spatial index structure.

From the experiments, we can conclude that multi-parent methods contribute to the prolonging of sensor network lifespan as long as child nodes find sufficient parent candidates. Among the three spatial index structures, OMSI-tree benefits most from the multi-parent methods. For the hybrid tree, which is improved from the nearest parent tree, multi-parent methods also show their strengths; network lifespan is almost two times longer. Overall, our experiments provide conclusive evidence that having alternative return paths is energy efficient and is able to prolong the lifespan of sensor networks.

3.3 Summary

In this chapter, our proposition is that a node should have multiple, alternative parent nodes on the path to the sinknode in order to distribute the workload of carrying query results back to the sink nodes; otherwise, the single-parent node will be overworked and loss its battery energy more quickly; thus, ending the lifespan of an effective sensor network more quickly.

We have conducted experiments with three different spatial index structures and vary the sensor network under different parameters such as sensor radio radius and sensor node density. Experimental results confirm our proposition that the multi-parent approach indeed extend the lifetime of sensor network on existing spatial index structures by sharing the workload of parent nodes. During running experiment, we also observed that spatial indices has very bad performance when facing large scale or high-density sensor network.

Chapter 4

Virtual Node Approach

When running the experiments for multi-parent methods, we found that none of the existing spatial index strategies is able to work well in high-density sensor network. By analyzing the topology of network, we noticed the reason is that high-density makes each node connects with more neighbors. The frequent communications among these redundant nodes exhaust the battery energy soon.

Various studies on node redundancy have been carried out. The idea of “magic number”, which was first proposed by Kleinrock and Silvester [30] to measure the average number of neighboring nodes required to keep the network connected, has been intensely discussed by several research groups [30, 25]. Depending on the protocols and assumptions, the “magic number” varies from five to eight. Other researchers pointed out that the existing “magic number” only works for small networks. For large scale networks, the “magic number” should be greater than $5.1774 \log n$ (where n is the number of total sensor nodes) to have the network asymptotically connected [58]. Besides the connection issue, coverage issue has also been studied. Gao *et al.* has theoretically analyzed the minimum and maximum number of neighbors that are required to provide complete redundancy [19]. Their conclusion is quite interesting: At least three neighbors and at most five neighbors are needed to make a node completely redundant. However, none of them has considered the side effects

Figure 4.1: Virtual grid and virtual nodes on sensor network

of high redundancy.

In this chapter, we first introduce two new concepts of redundancy; namely, *physical redundancy* and *logical redundancy*. Then we propose our *virtual node* approach to leverage on the advantages of both redundancies. The intuition for these two forms of redundancies are as follows. A node has a communication radius R and a sensing radius r , $r \ll R$. We place a virtual grid-cell on the sensor network where a cell's width and height are $r\sqrt{2}$, as illustrated in Figure 4.1. These cells are called atomic cells. After deployment, there might be more than one sensor nodes inside each atomic cell. It is very likely that sensor nodes inside the same atomic cell yield the same sensor reading as they have overlaps in the sensing area. In other words,

there is redundancy among the sensor nodes. We refer to this as *physical* redundancy. Depending on the area of a grid cell, it is also possible for sensor nodes within a group of nearby atomic cells to have overlapping sensing area. For instance, sensors inside the same meeting room are having the same temperature readings most of the time. There is also a form of redundancy here, which we refer as *logical* redundancy.

In both forms of redundancies, one may pick one sensor node to represent the rest of the nodes in terms of providing the sensor reading. That is, these nodes and the representative node form a *virtual* node. In a high-density network, a sensor node is able to communicate with up to tens or hundreds of nodes in its neighborhood. Due to the high communication cost, a sensor node may die quickly when it has to communicate with all its neighbors frequently. The virtual node approach groups all redundant nodes as one virtual node; one representative sensor node is active mode while the rest are in the sleep mode. Hence, communications are reduced and exist among active nodes only. Also, the total energy of sensor network is conserved.

The contributions of this chapter are as follows:

- We introduce the concepts of physical and logical redundancy. Motivated by the characteristic of both redundancies, we propose the idea of virtual node to leverage on the advantage of redundant nodes. By scheduling only one node to be in the active mode, sensor network lifespan is largely prolonged.
- We showed that using a combination of spatial index and virtual node, one is able to overcome the disadvantage of current spatial index approaches. Current spatial indices are not able to handle large scale and high-density sensor network in an energy efficient manner. To the best of our knowledge, this is the first work to handle both large scale and high-density sensor network with spatial indices.

- We have evaluated the performance of the proposed idea in terms of successful query evaluation and message latency, and compared them with the approach without virtual nodes. The experimental results showed that our virtual node approach is able to prolong sensor network lifespan especially when the virtual node size is large or node density is high.

4.1 Virtual Node Methods

In this section, we first provide some definitions and notations for the concept of redundancy in sensor networks. Then, we describe our proposed concept of virtual node and how one makes use of virtual nodes to reduce overall energy cost of the sensor network.

4.1.1 Node Redundancy

The following notations are used in this chapter:

- The set of sensor nodes in a sensor network is denoted by $\mathbb{S}$.
- For all the sensor nodes, we assume that they have the same sensing radius and communication radius, which are denoted as r and R respectively.
- Let $A(i, r)$ denote the sensing area covered by node i .
- Let $A(i, R)$ denote the communication area covered by node i . Generally, $r \ll R$ and $A(i, r) \subset A(i, R)$.

Definition 1 For a give node $i \in \mathbb{S}$, if $\exists j \in \mathbb{S}$, $i \neq j$, we have

$$A(i, r) \cap A(j, r) \neq \phi.$$

We say that node i is partially redundant.

Definition 2 For a give node $i \in \mathbb{S}$, if $A(i, r) \subseteq \bigcup_{j \in \mathbb{S}} A(j, r)$, where $j \neq i$. We say that node i is completely redundant.

Definition 3 For a give node $i \in \mathbb{S}$, its sensing neighborhood set is defined as:

$$SN(i) = \{j \in \mathbb{S} \mid A(i, r) \cap A(j, r) \neq \emptyset, j \neq i\}$$

Definition 4 For a given node i , its cell neighborhood set is defined as:

$$AN(i) = \{j \in \mathbb{S} \mid x_j = x_i \wedge y_j = y_i, j \neq i\}$$

where (x_i, y_i) is the ID of the cell node i belongs and (x_j, y_j) is the ID of the cell node j belongs. As there is not much difference between atomic cell size and $A(i, r)$, we may refer to the sensing neighborhood set as the atomic cell neighborhood; i.e., $AN(i) \approx SN(i)$. In addition, atomic cell neighbor also represents virtual node neighbor in physical redundancy case.

Definition 5 Given two nodes i and j , as long as $A(i, r) \cap A(j, r) \neq \emptyset$, we say that node j is physical redundant with respect to node i and vice versa. In other words, if $SN(i) \neq \emptyset$, node i is physically redundant with respect to other nodes. From the definition, we see that both partial redundancy and complete redundancy are special forms of physical redundancy.

Definition 6 Given two nodes i and j , $i \cap AN(j, r) = \emptyset$ that are likely to yield the same sensor values most of the time, node i is logically redundant with respect to node j and vice versa.

4.1.2 Virtual Node

In this section, we introduce the concept of virtual node with respect to *physical redundancy* (atomic cells) and *logical redundancy* (multi-cells).

4.1.3 Virtual Node Approach for Physical Redundancy

Consider a virtual grid superimpose on a sensor network and each sensor node's sensing radius is r , we have the following theorem:

Theorem 4.1 *For a node i , if $AN(i) \neq \phi$, node i is a physically redundant node.*

Proof: Figure 4.2(a) shows node i located at the center of an atomic cell. From the width of the atomic cell and sensor node's sensing radius, we note that the sensing area is the circumcircle of the atomic cell; i.e., the atomic cell is covered by only one sensor node as long as the node is located at the center of the atomic cell. Since node i is located at the center, if $AN(i) \neq \phi$, there is at least one more node inside the atomic cell. From the definition of physical redundancy, we conclude that node i is physically redundant.

Figure 4.2(b) shows a more general case. Assuming that $j \in AN(i)$, it is intuitive that inside the atomic cell, wherever j may be located, node i is physically redundant. Figure 4.2(c) illustrates an extreme case: Node i and its atomic neighbor j are on the diagonal vertices. We consider that node i is physically redundant for the following reasons: First, the two circles are still connected at tangent point k ; second, this extreme case is rare; third, we generally have $|AN(i)| > 1$; i.e., there will be other neighbors inside the atomic cell and they are unlikely to be at the extreme location.

□

From the above proof, we may think of an atomic cell as a *virtual node* representing all the physically redundant nodes within the atomic cell. In the rest of this section, a virtual node represents an atomic cell. The virtual node ID value is in fact the atomic cell ID which it represents. Any arbitrary sensor node within a virtual node can be chosen as a representative of the other nodes while the latter are placed

4.2.a: Two nodes are in the same position of virtual node

4.2.b: Two nodes are inside virtual node but not in same position

4.2.c: Two nodes are on the two furthest position of same virtual node

Figure 4.2: Possible node location inside an atomic-cell

in the sleep mode to conserve energy. To achieve this, there are three issues to be addressed:

The first issue in the virtual node approach for *physical redundancy* is deciding which node (also called *active* node) to represent the virtual node. We can choose the node that is closest to the center of the virtual node, as the closer the node is to the center, the better coverage it gives.

The second issue is how to select the active node. There are two situations:

- The first situation occurs after the sensor network is partitioned into atomic cells by a virtual grid. Given a partitioned sensor network, each grid cell (called atomic cell) has width $r\sqrt{2}$ and each atomic cell has a virtual coordinate (x, y) . In the beginning, all sensor nodes are in active mode. Each sensor node first determines to which virtual node it belongs by following equations:

$$x = x_i \bmod r\sqrt{2}$$

$$y = y_i \bmod r\sqrt{2}$$

where (x_i, y_i) is the coordinate of sensor node; (x, y) is the ID of virtual node to which it belongs. Partitioning in this way guarantees that each node belongs to exactly one virtual node. After all sensor nodes have found their virtual nodes, each sensor node sends a “hello” message that includes its location information to its virtual node (atomic cell) neighbors. If the receiver finds that the sender is closer to the virtual node center than itself, it switches itself into sleep mode. When the message exchange period is over, there is only one node in the active mode. Each active node represents the whole virtual node.

- The second situation occurs when an active node fails. Whenever an active node determines that its energy level is below certain threshold ε , it periodically sends a “goodbye” message to its virtual node neighbors. The rest of

the nodes will elect a new active node by repeating the process mentioned in the first situation. The new active node will send an acknowledgement to the previous active node. Once the previous active node receives an acknowledgement from any virtual node neighbor, it sends all the information it has to acknowledgement sender and switches itself into sleep mode. In this approach, we need to ensure that the active node is heard by its virtual node neighbors before its failure; otherwise, there will not be any further active node as the election token is missing. In order to guarantee this, all resting nodes need to periodically turn into the active mode to check on the active node's energy level ε via message exchange. If the current active node has not sent any "goodbye" message, they switch into the sleep mode again. To achieve energy efficiency, the wake-up period is not fixed but depends on how much energy the active node remains. There is no need to check the active node frequently when it still has a high level of energy.

The third issue is how to make the selected active nodes connected. In other words, how to make virtual nodes connected. Generally speaking, there are two approaches: One is to apply *ad hoc* protocols and the other one is to use spatial indices. As the objective of this work is to adapt spatial indices to large scale and high-density sensor network, we adopt the spatial index approach instead of *ad hoc* protocols.

We adopt three spatial indices to evaluate the performance of our virtual node idea. The three spatial indices were first proposed by Zha *et al.* [64]. We make modifications to these indices as follows: We introduce a new characteristic for virtual node: *strength*. Given that node i is an active node, the strength of a virtual node is the number of virtual node neighbors node i has, denoted as $|AN(i)|$; the more neighbors a virtual node has, the stronger the virtual node is.

One thing we need to emphasize here is that the spatial index is built from virtual nodes rather than actual sensor nodes. In a spatial index, virtual nodes are organized in a hierarchical manner where directly connected virtual nodes are in a parent-child relationship. Active nodes store information of parent and child nodes using their virtual node ID's instead of actual node ID's. Suppose virtual node B is the child of virtual node A in the spatial index and sensor node b is the current active node in virtual node B and sensor node a is the current active node in virtual node A . Any message from B to its parent A actually comes from sensor node b and reaches node a . However, the destination ID within the sending messages adopts the ID of virtual node A instead of the ID of sensor node a . Sensor node a will respond to any messages whose destination ID value is equal to the value of virtual node A 's ID. In other words, sensor node b is completely unaware of the existence of sensor node a and vice versa.

The advantage of building spatial index at the virtual node level is to reduce the overheads of maintaining the spatial index. In a spatial index, a parent maintains all its children location information. Any parent-child relationship changes will result in an update of the parent. The update of parents will further result in other updates of its parents. This update process will not stop until it reaches the root of the spatial index. If a spatial index is built at the actual node level, the spatial index of the entire sensor network needs to be updated every time a virtual node switches its active node. The cost is not trivial. While building a spatial index at the virtual node level, only virtual node neighbors need to know the changes made by switching active nodes. To the outsider, the switching of active nodes is not their business as long as the virtual node ID does not change. The only cost of maintaining the spatial index is the message exchange inside that virtual node. Minimizing the cost of maintaining spatial index is achieved in this way.

4.1.4 Virtual Node Approach for Logical Redundancy

A highly dense deployment of sensor network provides seamless coverage of a monitored area. However, in reality, we may not require seamless coverage as long as the sensor values of neighboring nodes do not vary much from one another. For instance, in a small size meeting room with four sensor nodes deployed at each corner to detect the average temperature, it is likely that the four sensors detect the same temperature values all the time. By Definition 6, we know that this is *logical redundancy*. Any of the logically redundant nodes may represent the other nodes to reduce total energy consumption. Our virtual node approach for *logical redundancy* is motivated by this characteristic.

Figure 4.3 illustrates the case of virtual node for logically redundant sensor nodes. The shadowed cells belong to the same virtual node. So far, we only consider regular shapes: square, and all the virtual nodes are of the same size. The number of atomic cells inside the same virtual node is n^2 where $n = 1, 2, \dots$. We note that the virtual node approach for *physical redundancy* is a special case where $n = 1$. Theoretically, it is possible for $n \rightarrow \infty$. However, due to the constraint of sensor node communication radius, the width of virtual node has to be constrained to guarantee that active nodes are connected.

The virtual node approach for *logical redundancy* predefines the size of virtual node to n and all virtual nodes are of the same size. The first issue to apply the virtual node approach for *logical redundancy* is group sensor nodes are grouped into virtual nodes. Similar to the virtual node approach for *physical redundancy*, virtual node ID is obtained as follows:

$$\begin{aligned}x &= x_i \bmod rn\sqrt{2} \\ y &= y_i \bmod rn\sqrt{2}\end{aligned}$$

Figure 4.3: Virtual node approach for logical redundancy

where $r\sqrt{2}$ is the atomic cell size, n is the virtual node size and $rn\sqrt{2}$ is the virtual node width. Sensor nodes inside the same virtual node adopt the same approach as for *physical redundancy* to select active nodes. We need to point out two things here: The *strength* of the virtual node approach for *logical redundancy* is equivalent to the total number of sensor nodes within that virtual node; not all sensor nodes in two nearby virtual nodes are able to detect each other, as illustrated in Figure 4.4. Nodes $v1$, $v2$ and $v3$ are three virtual nodes for *logical redundancy*; node i is the current active node in $v1$. The shadowed circle shows the communication range of node i . It is intuitive that active node i is not able to fully cover virtual nodes $v2$

Figure 4.4: Connection between virtual nodes

and $v3$. In other words, suppose virtual node $v2$ has active node j , virtual node $v1$ is able to observe virtual node $v2$ if and only if $j \in A(i, R)$. For virtual node $v3$, what happens is that $k \cap A(i, R) = \emptyset$. If all the virtual nodes around $v3$ are not able to detect the active node k , $v3$ is disconnected from network; we call this *fake death*. *Fake-death* node may reconnect to the network after its neighbors switched their active nodes. For instance, when virtual node $v1$ switches its active node from i to i' , $v3$ is detectable again. Although the *fake death* nodes are capable of rejoining the network, it is not desirable as this increases the cost of maintaining spatial indices. We study the maximal size of virtual node that could prevent virtual node from *fake death* in the next section.

4.1.5 Maximal Size of Virtual Node

To avoid *fake death*, we study the maximal size of virtual node that is able to keep virtual node away from *fake death* while retaining the maximal benefit of node redundancy. In order to ensure that a virtual node is 100% detectable, the active

Figure 4.5: Maximal size of virtual nodes

node of a virtual node should fully cover at least one neighboring virtual node. It is intuitive that covering the nearest neighboring virtual node yields the maximal virtual node size. Now, we analyze the maximal virtual node size that guarantees that an active node fully covers at least one neighboring virtual node wherever the active node is. As shown in Figure 4.5, there are nine virtual nodes having width d . Take the middle virtual node v_0 for example. For any active node inside it, to maximize the width d , the active node should at least cover the furthest vertex of the nearest neighboring virtual node because the communication radius R is fixed. Due to symmetry, we only need to discuss one eighth of the virtual node as shown in Figure 4.5. The rest are the same.

Suppose point p is an arbitrary active node inside triangle $\triangle boc$ within virtual

node $v0$. Virtual node $v2$ is the nearest virtual node to point p . Point a is the furthest vertex in $v2$ to point p . Point c is the center of virtual node $v0$. Points b and d are other two vertexes of virtual node $v2$. Point o is the middle segment bd . Considering active node has the maximal communication radius R , the problem becomes finding a position for p inside $\triangle boc$ where the width d achieves the minimal value $d_{\min}$. Active nodes are able to remain connected as long as the virtual node size does not exceed $d_{\min}/r\sqrt{2}$.

Segments ab and bc are both on the diagonal of the square. We can easily conclude that

$$\angle oba = \angle obc = \pi/4;$$

hence, $\triangle abc$ is a right-angled triangle. Referring to the width d of virtual node, we have

$$ab = d\sqrt{2}, bc = d\sqrt{2}/2$$

According to Pythagorean theorem, we have the following:

$$ac^2 = \left(d\sqrt{2}\right)^2 + \left(\frac{d\sqrt{2}}{2}\right)^2$$

$$ac = d\sqrt{\frac{5}{2}}$$

We find that when ac achieves its minimal value, d achieves its minimal value ($d = d_{\min}$). So the problem changes to one of finding a point p inside $\triangle obc$ that makes ac minimal. Segment ac is the hypotenuse. The answer is quite obvious that when point p is at point c ; $ac = ac_{\min}$. Thus, our conclusion is that when point p is at point c , $d = d_{\min}$.

According to Pythagorean theorem, we have the following equation:

$$R^2 = \left(\frac{3d_{\min}}{2}\right)^2 + \left(\frac{d_{\min}}{2}\right)^2$$

and we obtain that

$$d_{\min} = \frac{2R}{\sqrt{10}}$$

As mentioned in the previous section

$$d = nr\sqrt{2}$$

Now we have

$$\frac{2R}{\sqrt{10}} = nr\sqrt{2}$$

as n can only be an integer

$$n = \left\lfloor \frac{R}{r\sqrt{5}} \right\rfloor$$

From the above analysis, $n = \lfloor R/r\sqrt{5} \rfloor$ is the maximal virtual node size under the communication radius limitation R .

Analysis of Energy Conservation

We consider an evenly distributed sensor network with average density d to see how much network lifespan can be extended. Assume the network covers an area of size S , the total number of sensor nodes should be $d \times S$. As we know a virtual node with size n is a square with $\sqrt{2}nr$ units on a side, then the area of a virtual node is $2n^2r^2$.

As we see the virtual node size n is at most $\lfloor R/r\sqrt{5} \rfloor$. The minimum number of virtual node would be:

$$\frac{S}{2(\lfloor R/r\sqrt{5} \rfloor)^2r^2} = \left\lfloor \frac{5S}{2R^2} \right\rfloor$$

Since we assume sensor nodes are evenly distributed, the maximal number of sensor nodes would be:

$$\frac{d \times S}{\left\lfloor \frac{5S}{2R^2} \right\rfloor} = \left\lfloor \frac{2d \times R^2}{5} \right\rfloor$$

Figure 4.6: Performance without virtual nodes

At the ideal situation, only one sensor node in each virtual node would be in active state. Based on the maximal number of nodes in each virtual node, the network lifespan will be extended by at most $\lfloor 2d \times R^2/5 \rfloor$ times. The result reflects the fact that with virtual node approach, both higher node density and bigger virtual node will lead to longer network operational lifespan.

4.2 Experimental Results

In this section, we evaluate the performance of the proposed virtual node approach on both *physical redundancy* and *logical redundancy* with respect to different density and virtual node size. We use a Blade 32-node cluster, each CPU is a Dual Quad Xeon 3.0GHz, 8GB RAM for each node, running the Rock 4.2.1 task management system. The programs are coded in Java. We use the MicaZ node specification (2,000mAh battery, sending and receiving consume 140.91mW energy and 36mW when radio is off) for the sensor nodes. The sensor nodes are deployed in a 300×300 grid; the density of sensor nodes varies from 0.032 to 1.024.

4.7.a: OMSI tree

4.7.b: Nearest parent tree

4.7.c: Hybrid tree

Figure 4.7: Performance of virtual nodes

Query Set

We synthetically generate a set of 1 million sensor queries with the following distribution of query spatial coverage: 87% of the queries are querying less than 25% of the total network area; 11% of the queries are querying less than 50% of the total network area; 1.5% of the queries are querying less than 75% of the total network area; the rest 0.5% of queries are querying more than 75% of the total network area.

Virtual Node Failure

In the previous section, we define the characteristic *strength* for virtual nodes. Every time a sensor node within a virtual node fails, the *strength* of the virtual node decreases by one. When the strength of a virtual node is zero, we say that the virtual node has failed.

Sensor Network Failure

We determine that a sensor network is no longer accessible when 75% of all virtual nodes have failed or 1 million queries have been executed, whichever comes earlier.

Other Parameters

We vary the **node density** in the 300×300 units grid from 0.032 (low) to 1.024 (high). For each density, the sensor nodes are randomly created in the grid. Sensor communication radius is set to 10 units, which is a typical bluetooth transmit distance and sensing radius is set to 0.5 unit. All experiments are repeated 3 times and averages are taken for all values. We also vary the **virtual node size**: 1 (inadequate), 3 (adequate), 5 (excessive). Each experiment is performed for the three spatial indices as mentioned before.

Query is issued every 10ms in the simulation, so that 1 million queries takes 10000 seconds. Duty cycle of sensor node is set to 200α (where *alpha* is the percentage of

Figure 4.8: Average latency with different virtual node size and without virtual node

remaining energy) seconds, which means a sensor node will wake up at least every 20000 queries are executed. Energy level threshold ε is set to 25 percent of the full capacity.

Results

To demonstrate improvements arising from the use of virtual nodes, we first show performance without virtual node, as shown in Figure 4.6. Three spatial index approaches are adopted; namely, *OMSI-tree*, *Nearest Parent Tree*(NPT) and *Hybrid Tree* [64]. From Figure 4.6, we note that as density increases, performance of all three spatial indices decreases. None of these three spatial indices is able to successfully complete the evaluation of 100,000 queries. The reason is that all spatial indices considered only one sinknode and have not been designed for large scale, high-density sensor network. In particular, each parent node contains more offsprings in OMSI than the other two indices. This explains why OMSI only attained half the performance of the other two indices.

From the conditions $R = 10m$, $r = 0.5m$ and $n = \lfloor R/r\sqrt{5} \rfloor$, we note that virtual node size in our experiments should not be larger than 8. Actually, we are going to test $n \in [1, 5]$ to see the trend ($n = 0$ means that the virtual node approach is not applied). Experimental results are shown in Figure 4.7. Figures 4.7(a), (b) and (c) correspond to the three different spatial indices *OMSI tree*, *Strongest Parent Tree*, and *Hybrid Tree*. The *Strongest Parent Tree* used here is basically the same as the *Nearest Parent Tree* in an earlier work [64]. However, due to a uniformly partitioned network, a virtual node is quite likely to have two or more neighbors being the nearest parents. In this case, we compare the strength of these nearest parents and choose the strongest one. If there is only one nearest parent, *Strongest Parent Tree* is actually a *Nearest Parent Tree*.

There is no doubt that *OMSI* benefits a lot from the virtual node idea. As the sensor node density increases from 0.032 to 1.024, there is major improvement on the query times. In particular, when the virtual node size is large ($n = 4$ and 5), the improvement is obvious. We also note that under the same redundancy, as the virtual node size increases, more queries can be executed. In the best case, the virtual node approach extends network lifespan by at least 10 fold.

For the *Strongest Parent Tree* and *Hybrid Tree*, when a virtual node is large, the performance drops initially and then it increases again. By analyzing the topology of the sensor network, we find that in the sparse case, queries can hardly reach any node. In other words, there is no sensor node detected in most queries. We consider the performance drop part an unstable part and ignore it. The real performance is shown after network density increases to 0.064. From Figures 4.7(b) and (c) we see that *Strongest Parent Tree* almost performs the same as *OMSI Tree*. While combining the advantages of *Strongest Parent Tree* and *OMSI tree*, *Hybrid Tree* outperforms the others.

Another benefit from the virtual node approach is the reduction of the average transmitting latency to a sinknode. The virtual node approach reduces the number of hops in communication; thus, achieving a reduction in message delay. As shown in Figure 4.8, we note a huge reduction in message delay from *Strongest parent tree* and *Hybrid Tree* when the virtual node approach is applied. With regard to *OMSI Tree*, it is not much affected by the node density and virtual node size as a result of its characteristic.

From the above results, we note that the virtual node approach indeed yields major improvement in sensor network lifespan as well as message latency. There are two parameters deciding the performance: sensor node density and virtual node size. When we increase sensor node density from around $0.032 \text{ nodes}/m^2$ (node number 2,880) to $1.024 \text{ nodes}/m^2$ (node number 92,160), the network shows its capability of satisfying much more queries. When we further increase the size of the virtual node, the improvement is significant. In the best case, the network is able to successfully complete 1 million spatial queries and continue to stay alive thereafter. In addition, the virtual node approach reduces message latency. From analysis, we know that this is due to the virtual node approach reducing the number of hops needed to deliver a message to a sinknode.

From the experiments, we may conclude that virtual nodes contribute to the prolonging of sensor network lifespan as long as the network is not sparse. The virtual node approach yields significant improvement when existing spatial indices are used. The denser the sensor network is and the larger the virtual node size is, the better performance it has. In the best case, network lifespan is almost tens times longer. Overall, our experiments provide conclusive evidence that putting the redundant sensor nodes to sleep while leaving only one node active to maintain network connectivity is energy efficient and is able to prolong the lifespan of the sensor network.

4.3 Summary

In this chapter, we have proposed the concepts of *physical redundancy* and *logical redundancy*, which led us to the idea of virtual nodes to leverage on the advantages of node redundancy. Our virtual node approach switches redundant nodes to sleep while leaving only one node active mode to reduce the energy consumption of the whole network. Due to the limitation of sensor communication capability, we have also theoretically analyzed the maximal size of virtual nodes. We have conducted experiments with three existing spatial indices and varied the sensor network density and virtual node size. Experimental results confirm our proposition that the virtual node approach indeed extends the lifespan of sensor network on existing spatial indices by sleeping the redundant nodes.

Chapter 5

Geographic Routing in Duty-Cycling Sensor Networks

Spatial index provides the optimal path for routings. However, the cost of maintaining a spatial index is not trivial. Whenever a node detects an update of its children, it updates its parent and this process is repeated until the update reaches the root of this index. Each update may cause a network wide traffic. This is highly inefficient, especially in large-scale networks. Geographic routing [6, 16, 17, 28, 31, 33, 34, 20], on the contrary, has received much attention due to the advantage of scalability.

Despite existing geographic routing algorithms differing on how to make face change decisions, most existing geographic routing algorithms are designed for static always-on sensor networks where all the sensor nodes are active all the time. In recent sensor network development, to increase energy efficiency and prolong network lifetime, sensor nodes are usually configured to duty-cycling work mode, in which not all sensor nodes are awake and available at all times. At certain time slots, only a portion of the network is available to send and receive packets [60, 44, 18].

To the best of our knowledge, none of the existing geographic routing algorithms guarantees the success of packet delivery in duty-cycling wireless sensor networks. For instance, in the face routing stage, some algorithms require packets to traverse the entire face so as to decide which sensor node on the face is closest to the destination

and the packets will then be delivered to that particular sensor node for the face change. In duty-cycling sensor networks, however, any individual sensor node may fall asleep after being selected as such a critical node. As a result, the data packet may miss the face change point and keep traversing on that particular face. After the TTL of the packet counts down to zero, the packet will then be dropped, which leads to transmission failure.

In this chapter, we study the problem of achieving reliable geographic routing in duty-cycling sensor networks. In particular, we fundamentally investigate the causes of the failures in the existing geographic routing algorithms that cannot guarantee the packet delivery in duty-cycling wireless sensor networks. We formally analyze and classify possible situations that lead to packet delivery failures. In particular, the sleep and wake-up operations of each sensor node can be mapped to the *face merge* and the *face split* phenomena in the original complete network topology. We study the topology change caused by duty-cycling mode and its impact on the geographic routing performance. After in-depth studying of the issues of geographic routing introduced by duty-cycling sensor networks, we propose a general algorithm component to serve existing geographic routing algorithms to guarantee the success of packet delivery in duty-cycling sensor networks. Both theoretical analysis and trace-driven simulation experiments are conducted to verify the effectiveness and efficiency of the proposed protocol. So far, this work is the first attempt to extend reliable geographic routing to duty-cycling sensor networks.

5.1 Duty-cycling Sensor Networks

In duty-cycling sensor networks, nodes alternate between sleep and wakeup modes to conserve energy and extend the network lifespan [29, 11, 45, 54, 21, 62]. Roughly, we

can divide duty-cycling sensor networks into synchronous and asynchronous duty-cycling sensor networks. Within synchronous duty-cycling sensor networks, all sensor nodes are simultaneously in the active state. The advantage is that sensor nodes do not have to wait for their neighbors to wake up to send the packet. Nevertheless, the disadvantage is the high overheads and impracticality in large scale sensor networks. Within asynchronous duty-cycling sensor networks, sensor nodes sleep or wake up on their own clocks. It is more practical and energy efficient. Therefore, asynchronous duty-cycling sensor networks are more popular. However, scheduling packet delivery is quite a challenging issue.

5.2 Motivations

Geographic routing algorithms reviewed in Chapter 2 utilize various criteria to make face change decisions. However, they all need to find an edge intersecting with the virtual line st , where s and t represent the source and the destination. We note that none of the existing protocols takes the network topology change into consideration in making a face change decision. Apparently, face routing algorithms that require complete face information to make face change decision are more likely to suffer delivery failure in duty-cycling network. The stateless algorithms, such as GPSR and GFG are less likely to suffer delivery failure due to their simple and stateless face change strategies. If GPSR and GFG fail to deliver in certain situations, it is highly possible that similar failures could occur in other algorithms as well. In the rest of this chapter, we choose GPSR as the vehicle to study delivery failure caused by node sleep and wakeup operations. Moreover, we will later show that such an analysis can be easily extended to other protocols. We observe several cases where the original face routing operations of GPSR fail in duty-cycling sensor networks.

5.1.a: Case 1

5.1.b: Case 2

5.1.c: Case 3

Figure 5.1: Observation of delivery failures

Case 1: Assume v_2 and v_5 in Figure 2.16 are asleep initially. Packet sent from s will arrive at v_6 following path s, v_1, v_3, v_4, v_6 as illustrated by red dash arrows in Figure 5.1.a. If v_5 then wakes up, v_6 forwards the packet to v_5 and v_5 will forward the packet back to v_3 enforced by the right hand rule in face routing. Since the packet cannot encounter an intersection with st in this example, it will keep looping in v_3, v_4, v_5, v_6 as illustrated by blue solid arrows. Therefore, delivery failures may occur since packet will finally be dropped during such an infinite loop.

Case 2: In Figure 2.16, packet enter F_3 and record p_3 as the entrance point. Assume v_{17} and v_8 go to sleep. The packet will arrive at v_{10} through path $v_{12}, v_{11} \dots v_{10}$. If v_8 wakes up at this moment and no more topology change happens, packet will keep traversing in F_2 because they cannot find an edge intersecting with line st closer to the destination than p_3 as illustrated by blue solid arrows in Figure 5.1.b.

Case 3: Assume v_{12} and v_{18} in Figure 2.16 are in sleep mode all the time. According to the right hand rule, packet from s will arrive at v_{21} through path $s, v_1 \dots v_{16}, v_{17}, v_{21}$, as shown by red dash arrows in Figure 5.1.c. If v_{15} goes to sleep at this moment, v_{16}, v_{17} and v_{21} become disconnected from the destination and transmission loops occur. In Case 3, there is no valid path to the destination, since v_{16}, v_{17} and v_{21} are isolated.

Based on the aforementioned cases, we find that packets delivery failures are mainly caused by topological loop and disconnection in duty-cycling wireless sensor networks. In the case of a loop, there exists a feasible path in the network to deliver packets to the destination. However, packets are stuck in the loop and fail to explore such a path with the existing geographic routing strategies. On the contrary, there is no feasible path at all in Case 3 and it is not possible to heal such a delivery failure. Therefore, in the rest of this chapter, we focus on recovering from the delivery failure caused by loops due to node wakeup and sleep.

5.3 Analysis of Delivery Failure

In this section, we formally analyze packet delivery failures by mapping node sleep and wakeup operations to a set of atomic network topology changes.

5.3.1 Atomic Network Topology Changes due to Node Sleep and Wake-up

Geographic routing is normally employed on top of a planarized topology. Existing works usually planarize network in to *Gabriel Graph*(GG) or *Relative Neighborhood Graph*(RNG). Their detailed construction process can be found in the related works [28, 6]. We omit it here due to space limitation. In a planarized network, wakeup and sleep operations result in a local connection and disconnection respectively. From the topological view, we can observe a sequence of face merges and splits happening in duty-cycling sensor networks. In order to systematically study the effects of sensor duty-cycle on network topology change, we map the sleep and wakeup operations to the face merge and split. Face merge and split are treated as atomic network topology changes.

Note that GG has a special property, i.e., an edge can exist between two nodes only if there is no other node inside the circle of that edge. Due to this unique characteristic, node sleep or wakeup may result in both adding and removing of edges, which could in turn cause more than one atomic network topology change simultaneously. We will analyze topology changes caused by sleep and wakeup operations, and further study all possible delivery failures caused by each topology change.

Face merge caused by node sleep. For the example shown in Figure 5.2, there are three faces F_2 , F_5 and F_6 initially. They share edges $v_{11}v_8$, v_7v_8 and $v_{14}v_8$ with each other. When node v_8 goes to sleep, v_{11} , v_7 and v_{14} notice that they cannot reach v_8 any more, so connections $v_{11}v_8$, v_7v_8 and $v_{14}v_8$ disappear. Additionally, those nodes

5.2.a: Node v_8 is awake, three faces are in the network

5.2.b: Node v_8 goes to sleep, three faces merge into one face

Figure 5.2: Example of face merge and split

will check if any new connection can be established with their one-hop neighbors due to the absence of v_8 . The new topology is shown in Figure 5.2.b. F_2 , F_5 and F_6 merge into a new $Face(v_2 \dots v_{10})$.

Face merge and split caused by node sleep. These two atomic network topology changes happen simultaneously. Refer to Figure 5.4.a for example. v_{14} is located in the circles of edge v_8v_{16} and edge v_7v_{15} . The disappearance of v_{14} not only removes all the edges incident to itself, but also brings new edges which are unable to exist due to the existence of v_{14} , as illustrated by Figure 5.4.b, namely v_8v_{16} and v_7v_{15} .

Network disconnection caused by node sleep. Disconnection happens when a network is split into more than one part, as illustrated by Figure 5.1.c. There is no path to connect nodes of different parts. Nodes v_{16} , v_{17} and v_{21} are not able to connect to the rest of the nodes.

Face split caused by node wakeup. Face split is the reverse process of a face merge.

An example of face split can be given by referring to the example of face merge in a reverse manner. As shown in Figure 5.2, v_8 wakes up and establishes connections with all its one hop neighbors as long as there is no other node inside that connection circle. In addition, all the nodes connecting to v_8 check if they should remove any edge due to the appearance of v_8 . Eventually, $Face(v_2...v_{10})$ is split into F_2 , F_5 and F_6 .

Face split and merge caused by node wakeup. These two atomic network topology changes also happen in the same transaction. Refer to Figure 5.3.a for example, v_9 is inside the circle of edge v_2v_5 . If v_9 wakes up, edge v_2v_5 must be removed due to the characteristic of GG and new edges v_2v_9 , v_9v_8 and v_5v_9 will be added as shown in Figure 5.3.b.

Network connection caused by node wakeup. Connection appears when a node wakes up and establishes connections with two or more disconnected parts. Refer to Figure 5.1.c for example. If v_{15} wakes up, paths from those disconnected nodes (v_{16} , v_{17} and v_{21}) to the destination appear.

5.3.2 Delivery Failure due to Packet Leaving Major Face

We define a *major face* as the face intersecting with the virtual line st . In traditional geographic routing algorithms, packets traverse along major faces all the time and are guaranteed to be received by the destination. However, in duty-cycling networks, a packet may not always traverse in major faces. In a non-major face, the packet cannot meet an edge intersecting with line st and thus is unable to make a face change decision. We study all the possible topology changes that lead to packets leaving major faces.

Delivery failure caused by face split (node wakeup)

Refer to Figure 2.16, a packet is sent from s to t . Initially, the packet stays on the major face F_1 . By right hand rule, s forwards the packet to v_1 . At this moment,

v_2 goes to sleep. Edge v_1v_2 , $v_{10}v_2$ and v_5v_2 disappear. Faces F_2 and F_7 merge into a major face. Then, v_1 forwards the packet to v_3 instead of v_2 . When this packet arrives at v_3 , v_5 goes to sleep. Face F_8 merges into a major face. v_3 forwards the packet to v_4 and v_4 forwards it to v_6 . If v_5 wakes up at this time, face F_8 is split from the major face. v_6 decides to forward the packet to v_5 . Thereby, this packet is stuck in face F_8 with loop v_3, v_4, v_6, v_5, v_3 as illustrated by blue solid arrows in Figure 5.1.b, namely, packet delivery fails.

Delivery failure caused by face split and merge (node wakeup)

Face split and merge may happen in the same transaction due to the characteristic of building a planar graph. When a sleeping node is located inside the other edge's circle, its appearance not only brings in new connection, but also removes some existing connection.

We refer to Figure 5.3 as an example of packet delivery failure caused by face split and merge. Suppose all the nodes are awake except v_9 and v_{10} . Through a sequence of forwarding, packet reaches v_7 as illustrated in Figure 5.3.a. At this moment, v_9 wakes up. Connection between v_9 and its one hop neighbors are established, namely, v_2v_9 , v_5v_9 and v_8v_9 . $Face(v_5v_6v_7v_8v_9)$ is split from the major face, so is $Face(v_5v_9v_2)$. In addition, because v_9 is located inside the circle of edge v_2v_5 , the connection between v_5 and v_2 is removed due to the planar graph construction with GG. $Face(v_5v_9v_2)$ merges with the $Face(v_1v_3v_5v_2)$ into a larger face, $Face(v_1v_3v_5v_9v_2)$. Under the current circumstance, v_7 forwards the packet to v_8 by the right hand rule. Thus, packet loops in path v_7, v_8, v_9, v_5, v_6 as illustrated by blue solid arrows in Figure 5.3.b. Note that, in this example, removing v_2v_5 or not does not change the fact that a delivery failure occurs. In other words, delivery fails due to face split only though a face merge happens as well.

Delivery failure caused by face merge and split (node sleep)

Similar to the previous case, face merge and split may also happen as the same transaction. An example is shown in Figure 5.4. v_{14} is located inside the circles of v_8v_{16} and v_7v_{15} . There is no connection between v_8 and v_{16} , as well as between v_7 and v_{15} . Assume face F_3 and F_5 are merged into a major face before s sending out a packet, meaning v_{12} is asleep at the beginning. Packet sent from s can successfully reach v_{15} through path $s, v_1, \dots, v_{14}, v_{15}$ as illustrated by red dash arrows in Figure 5.4.a. Upon v_{15} receiving the packet, v_{14} goes to sleep. Face F_6 and the outer face are merged into a major face. Since v_{14} no longer exists, edge v_8v_{16} and v_7v_{15} appear and split $Face(v_8v_7v_{15}v_{16})$ from the major face. By the right hand rule, v_{15} forwards the packet to v_{16} , v_{16} forwards the packet to v_8 , v_8 forwards the packet to v_7 and v_7 forwards the packet to v_{15} . From v_{15} , the packet starts looping from v_{15} as illustrated by blue solid arrows in Figure 5.4.b.

From above analysis, we note that topology changes in the form of face split, face merge&split and face split&merge may cause a packet to leave a major face and which further leads to delivery failure. We also note that when the topology changes in the form of face merge&split or face split&merge, the essential cause of the delivery failure is face split.

Lemma 5.1 *Packet which is currently on a major face may be moved to a non-major face due to face split.*

Proof: Suppose the set of nodes on a major face before a face split is V and after face split is V' . Then, all the nodes in $V - V'$ are not on the major face after the face split. When a face split happens, the packet could be on any node, say w . If $w \in V'$, packet is still on the major face. Otherwise, it is on a non-major face. $\square$

Lemma 5.2 *Delivery fails due to a loop if packet is on non-major face and there is no node incident to an edge intersecting with line st on that face.*

Proof: By right hand rule, packet only traverses its current face. A face change decision is made when a packet arrives at a node which is incident to an edge intersecting with line st . A non-major face does not guarantee that this kind of node exists. Once that kind of node cannot be met, packet keeps looping on that non-major face and delivery fails. $\square$

5.3.3 Delivery failure while packet is still on major face

We denote the major face as F_k . In geographic routing, a packet enters F_k from $F_k.in$ and leaves from $F_k.out$ in a static sensor network. In duty-cycling sensor network, even if the packet stays on a major face all the time, the success of delivery is not fully guaranteed because the major face it currently traverses may not be F_k . In other words, packet enters major face F_k but currently may traverse major face F'_k , where $F'_k \neq F_k$. Face change decision cannot be made if $|tF'_k.out| > |tF_k.in|$, where $|tF'_k.out|$ is the Euclidean distance between t and $F'_k.out$, $|tF_k.in|$ is the Euclidean distance between t and $F_k.in$.

Lemma 5.3 *Packet enters face F_k from $F_k.in$. Due to topology change, F_k changes to F'_k . Delivery fails if there is no node in F'_k closer to the destination than $F_k.in$.*

Assume the packet enters major face F_k , but due to topology change, it currently traverses on a major face F'_k . For the sake of simplicity, we use a dotted line with gray nodes to represent F_k and a solid line with black nodes to represent F'_k . All the possible F_k and F'_k relations are illustrated by Figure 5.5.

Take Figure 5.5.a for instance, face F'_k is inside of F_k . That could be caused by a face split. Under current circumstance, $|tF_k.in| > |tF'_k.in| > |tF'_k.out| > |tF_k.out|$. When packet arrives at $F'_k.out$, it detects $F'_k.out$ is closer to destination than $F_k.in$. Then, a face change decision is made.

Similarly, we note that $|tF_k.in| > |tF_k.out| > |tF'_k.in| > |tF'_k.out|$ from Figure 5.5.b, $|tF_k.in| > |tF'_k.in| > |tF_k.out| > |tF'_k.out|$ from Figure 5.5.c, $|tF'_k.in| > |tF_k.in| > |tF'_k.out| > |tF_k.out|$ from Figure 5.5.d and $|tF'_k.in| > |tF_k.in| > |tF_k.out| > |tF'_k.out|$ from Figure 5.5.e. For all these five situations, $|tF'_k.out| < |tF_k.in|$. There is no problem for the packet to make a face change decision.

However, for the situation illustrated in Figure 5.5.f, $|tF'_k.in| > |tF'_k.out| > |tF_k.in| > |tF_k.out|$. A face change decision cannot be made due to $|tF_k.out| > |tF_k.in|$. An example of this situation is illustrated in Case 2 in Figure 5.1.b. We review it from the view of topology change. When a packet reaches v_{12} and detects $v_{12}v_{13}$ intersects line st at p_3 , it makes a face change decision and enters face F_3 . After that, the network topology changes, namely, F_3 , F_5 and F_2 merge together. By the right hand rule, packet reaches v_{10} from v_{12} . Again, the network topology changes, namely, F_2 is split. Apparently, both p_1 and p_2 are further to the destination compared to p_3 . Face change decision cannot be made on F_2 .

From the above analysis, we note that packets on a major face are less likely to fail to make a face change decision. Among all the possible new major face positions, only one situation will cause delivery failure. In the next section, we propose an algorithm which works for both delivery failures on non-major face and major face.

5.4 Algorithm Design

The major reason of delivery failure is that the packet cannot make face change decision, either because of a loop on a non-major face or $|tF'_k.out| > |tF_k.in|$. On the one hand, traversing on a non-major face, a packet definitely has no chance to make a face change decision as the face does not intersect with the virtual line st . On the other hand, traversing on major faces, packet is deliverable most of the time as long as the condition $|tF'_k.out| < |tF_k.in|$ is satisfied. Motivated by this observation, we

propose an algorithm to detect whether a packet is traversing on a major face and redirect the packet back to the major face if it is true. In addition, if a packet cannot meet $|tF'_k.out| < |tF_k.in|$ on the current major face, it will be forwarded to a new face. Such a forwarding will be repeated until $|tF'_k.out| < |tF_k.in|$ is satisfied.

The basic idea of our algorithm is to record the regular path traversed in the current face along with each packet. By “regular”, we mean that no loop has been detected yet on the path. Whenever a node receives a packet, it can refer to the record to check if a loop exists and based on that information make the next hop decision. We say there is a loop whenever a node receives the same packet more than once but the forwarding decision does not change.

Some notations and functions are declared in order to clearly describe our algorithm.

- $Q(a, b, \dots)$, where Q records the regular path of the current face so far traversed by the packet. It is represented by a series of node IDs. The nodes visited earlier are stored towards the head. Such a queue is carried along with the delivered packet.
- e_{in} is the incoming link of a packet. e'_{in} is the reverse of an incoming link. e^x_{in} is the incoming link when node x was recorded by Q .
- e_{out} is the current outgoing link. e'_{out} is the reverse of an outgoing link. e^x_{out} is the outgoing link when node x was recorded by Q .
- $R((e_{in}), G^x)$, where G^x is the network topology when x is recored by Q , generates outgoing link based on incoming link and network topology.

When s is sending out a packet, it creates a queue in that packet to store the regular path that the packet traversed in addition to source s and destination t .

Algorithm 3 Redirect packet back to major face

```

1: when node  $u$  receives a packet with header  $(s, t, Q)$  along edge  $e_{in} \leftarrow wu$ , where
    $s$  is the packet source,  $t$  is the destination, and  $Q$  stores all the visited nodes in
   current face by this packet. It performs the following process:
2: read  $s, t, Q$  from header and add  $u$  to  $Q$ 
3: if  $u = t$  then
4:   delivery ends successfully
5: else if  $t$  is neighbor of  $u$  then
6:    $e_{out} \leftarrow ut$ 
7:   send packet through  $e_{out}$ 
8: else
9:    $e_{out} \leftarrow$  the first link clockwise around  $u$  from  $e_{in}$ 
10:  if  $e_{out}$  intersects line  $st$  then
11:     $e_{out} \leftarrow$  the first link clockwise around  $u$  from  $e_{in}$  which does not intersect  $st$ 
12:     $Q = \{u\}$ 
13:  else if  $e_{out}$  intersects  $Q$  then
14:     $e_{out} = e'_{out}$ 
15:     $Q = u$ 
16:  else if loop is detected then
17:    for each node  $x = u$ , where  $x \in Q$  do
18:      if  $R((e_{in}^x), G^x) \neq R((e_{in}^x), G^u)$  then
19:         $e_{out} = R((e_{in}^x), G^u)$ 
20:        break
21:      end if
22:    end for
23:  end if
24:  send packet through  $e_{out}$ 
25: end if
    
```

Once the packet meets the condition to change face, the stored information can be removed because visiting the next major face is independent of the previous major face and the packet always moves to a face closer to the destination than the previous face. This property makes our algorithm scalable.

Details of our algorithm are described by Algorithm 3. Generally, upon receiving a packet, node u decides the outgoing link e_{out} by the following phases:

Phase 1: Node u first checks if it is the destination. If yes, the packet is delivered successfully. Otherwise, go to phase 2.

Phase 2: If node u detects the destination t is in its one-hop neighbor list, it forwards the packet through the outgoing link $e_{out} \leftarrow ut$. Otherwise, go to Phase 3.

Phase 3: Node u sets the first link clockwise around itself from e_{in} as a candidate outgoing link. If this candidate link intersect st , node u sets the reverse way of candidate outgoing link as e_{in} and sets e_{out} as the first link clockwise around u from e_{in} that does not intersect st . Node u resets $Q = u$ and forwards the packet through e_{out} . Otherwise, go to phase 4.

Phase 4: Node u checks the candidate outgoing link decided in phase 3 against the path recorded by Q . If an intersection is found, the outgoing link is reversed $e_{out} \leftarrow e'_{out}$ and the queue is reset to $Q = u$. Node u forwards the packet through the outgoing link. Otherwise, go to phase 5.

Phase 5: Node u checks if there is a loop by appending itself to Q . Once a loop is detected, node u re-examines the outgoing link made previously. If a different outgoing link can be made under the current network topology, node u forwards the packet through that outgoing link. Otherwise, the packet is forwarded through candidate outgoing link decided in phase 3.

We take the case that a loop is caused by a face merge&split case as an example to show how our algorithm assists GPSR to avoid a loop. To distinguish a node from its recorded instance, we use a' to represent the recorded instance of node a . When a packet arrives at v_8 for the second time, node u compares $R(e_{in}^{v'_8}, G^{v'_8}) = e_{v_8 v_{14}}$ with $R(e_{in}^{v'_8}, G^{v_8}) = e_{v_8 v_{16}}$ and finds the difference. v_8 resets queue to $Q = (v_8)$ and forwards the packet through the outgoing link $e_{v_8 v_{16}}$. A loop is avoided. Packet delivery path under our algorithm is shown in Figure 5.6.a.

Take a delivery failure on major face as another example. When v_{11} receives a packet from v_8 , illustrated by arrow 9 in Figure 5.6.b, it finds itself recorded by Q already. Therefore, v_{11} checks whether $R(e_{in}^{v'_{11}}, G^{v'_{11}})$ equal to $R(e_{in}^{v'_{11}}, G^{v_{11}})$. The

result is negative so v_{11} resets queue to $Q = (v_{11})$ and forwards packet through $R(e_{in}^{v'_{11}}, G^{v_{11}})$, namely, through the outgoing link $e_{v_{11}v_8}$. v_8 receives the packet and forwards it to v_{14} . Until now, a loop in F_2 has been avoided. The packet is able to meet a node with an edge intersecting line st on F_5 . The exact packet delivery path is illustrated by Figure 5.6.b.

5.4.1 Algorithm Properties

The key function of our algorithm is to detect whether a packet is unable to make a face change decision on the current face. If so, the algorithm will redirect it to a new face for face routing. Such a process will be repeated until a face change decision has been made.

There is no loop on a major face if the packet can successfully make a face change decision. Therefore, whenever a node receives the same packet twice and the next hop node is still the same, the packet must have failed to make a face change decision. To redirect the packet back to a new face, the first step is to detect whether the packet is unable to make a face change decision on the current face, namely, detect a loop.

Lemma 5.4 *With the information recorded by Q , a node is able to detect loop.*

Proof: We define there is a loop if a node receives the same packet and forwards it to the same next hop node. Upon receiving a packet, a node u can easily detect if this packet has been received before by checking Q . By the right hand rule, the next receiver can be decided by $R(e_{in}^{u'}, G^u)$. The former outgoing link can be found from Q . Thus, there is sufficient information for node u to detect loop. $\square$

Once it is confirmed that a packet is unable to make a face change decision on the current face, the next step is to redirect it back to a new face. Our algorithm redirects the packet to a face which has been partially visited before.

Theorem 5.2 *Repeat redirecting packet to a new face whenever a loop is detected. Packet will finally reach a face where face change decision can be made.*

Proof: When a loop is detected, the packet must reach a previously visited node a . Packet forwarding decision of a must be different under current topology otherwise there is a loop. If a new forwarding decision still meets a loop, a node b which was visited earlier than a will make the new forwarding decision. By repeating this process, the packet can finally get back to the node from which it entered current face. $\square$

The proposed algorithm component helps to route the packet back to the major face to avoid delivery failure with the cost of recording extra information. Comparing it with the original geographic routing algorithms, the overheads of our algorithm component is slightly higher due to recording of the traversed path within the current face. Such an overheads will not reduce the performance of the algorithm much because it requires a few bits to record a node ID and the quantity of nodes on a face is usually small. In addition, our algorithm cleans useless information whenever the packet starts exploring a new face. During the whole packet delivery procedure, the average overheads is quite low.

5.4.2 Delivery Guarantee

Generally, it is impossible to unconditionally guarantee delivery in a duty-cycling sensor networks. For example, there is no physical path to the destination due to duty-cycling. Geographic routing algorithms do not make duplicate copies of a packet. We cannot expect it to resend the packet when the destination is reconnected to the network. In this part, we define a set of conditions under which our algorithm is able to guarantee packet delivery.

Condition 1: A node cannot go to sleep after receiving a packet until the packet has been forwarded.

Usually, geographic routing algorithms have no idea of sensor node duty cycle. It may consider deliver failed if a node receives a packet and goes to sleep, though that node may wake up and forward the packet later. Discussion of sensor node duty cycle pattern is out of scope here. $\square$

Condition 2: There is a stationary spanning graph that remains connected during each face traversal.

Let t_i be the time a packet enters the i th face, where $i \geq 1$. We denote the network topology at time t_i by $G(t_i) = \{V(t_i), E(t_i)\}$, where $V(t_i)$ is the set of nodes awake at time t_i and $E(t_i)$ is the set of exiting links at time t_i . The stationary spanning graph of $G(t_i)$ is denoted by $G'(t_i, t_i + \Delta t) = \{V'(t_i, t_i + \Delta t) + E'(t_i, t_i + \Delta t)\}$, where Δt is the time takes to start next face traversal or reach the destination, $V'(t_i, t_i + \Delta t) = \{v | v \in V(t'), t' \in [t_i, t_i + \Delta t]\}$ is the set of nodes that remain awake during period Δt and $E'(t_i, t_i + \Delta t) = \{e | e \in E(t'), t' \in [t_i, t_i + \Delta t]\}$ is the set of links that remain active during period Δt . $\square$

Condition 3: Every node in the network performs at most one state transfer during each face traversal.

This condition limits each node $v \in (V(t_i) - V'(t_i, t_i + \Delta t))$ to doing one state transfer only. A node can either transfer from sleep to wakeup or from wakeup to sleep but cannot transfer back to its initial state. Note that, this condition only slightly reduces the scenarios to which our algorithm can be applied. That is because the packet forwarding speed and packet's state change frequency are not in the same order of magnitude. In reality, it is unlikely a node can change its state more than once before a packet is delivered. $\square$

Note that Δt is the time it takes to traverse one face. It is not necessary for the spanning graph to remain stable and connected from the beginning to the end,

because traversing the next face is independent of the previous face. As long as these three conditions are satisfied, our algorithm is able to guarantee delivery.

Theorem 5.3 *Given a network with n sensor nodes, there is a time $\Delta t \leq 2(n^2 - n)$. For all $i \geq 1$, our algorithm is able to guarantee packet delivery if the subgraph $G'(t_i, t_i + \Delta t) = \{V'(t_i, t_i + \Delta t) + E'(t_i, t_i + \Delta t)\}$ remains stable and connected.*

Proof: The largest face in any n nodes graph consists of $(n - 1)$ links. In the worst case, a packet may traverse both interior and exterior faces before finding a new face. Assuming each packet forwarding takes 1 time unit, the upper bound of traversing a face in a stationary graph is $2(n - 1)$. Suppose there are m nodes that change their states during a face traversal, namely, $|V(t_i) - V'(t_i, t_i + \Delta t)| = m$. In the worst case, each state change of a node causes the packet to revisit the same face from the beginning. The total time it takes to complete the current face traversal is at most $m \times 2(n - 1)$. Since $m \leq n$, it takes less than $2(n^2 - n)$ time to finish one face traversal.

Given a pair of source and destination nodes, there are limited links that intersect source-destination line in an n nodes graph. In other words, there are limited faces to traverse before reaching the destination. The total time costs to deliver a packet is bounded at $O(2(n^2 - n))$. Therefore, our algorithm is able to deliver packet in a limited time. $\square$

5.5 Experimental Results

To evaluate the performance of our proposed algorithm, we simulated a sensor network with a real sensor network topology, *greenOrbs*[38], which includes 444 sensor nodes with a maximal communication radius $100m$ deployed in a $200 \times 600m^2$ area. A time slot is set to 1 second and the working period T of each node is 100 time

slots. A 10% duty-cycling network means each node randomly picks up 10 time slots to wake up. Their duty-cycle patterns may be different but the total active time slots are the same. As we discussed in Chapter 2, GPSR is less likely to fail in duty-cycling sensor network among all the highlighted geographic routing algorithms. The simulations mainly focus on the comparison between GPSR and GPSR with our algorithm component.

5.5.1 Improvement of Proposed Algorithm Component in Duty-cycling Sensor Network

We run both GPSR with and GPSR without our algorithm component with different duty-cycle rate ranges from 10% to 100%. Four sets of experiments have been conducted in Figure 5.7 via varying the number of source-destination pairs from 100 to 400. In each experiment, the packet delivery success rate is illustrated.

Tests with 4 different source-destination sets yield similar results. With extremely low duty-cycle, e.g., 10% to 30%, either with or without our algorithm, GPSR has very low success rate. From 40% active duty-cycle, GPSR with our algorithm shows an advantage in the success rate. With an 80% active duty-cycle, the difference of success rate reaches the maximal value. With our algorithm, GPSR is able to deliver almost 80% of the packets. However, running GPSR alone it is only able to deliver 15% of the packets. When running at 100% duty-cycle, either with or without our algorithm, GPSR can deliver all the packets.

As we have analyzed, delivery failure in GPSR is caused by loops and disconnection. With the assistance of our algorithm, delivery failure is caused by disconnection only.

5.5.2 Loops in Duty-cycling Sensor Network

Success rate experiments indicate that loops are most likely to appear at 80% active duty-cycle. We recorded all the loops when GPSR (without our algorithm) was running with 80% active duty-cycle and visualized them in Figure 5.8.

As we can see, most of the loops are within 3 or 4 nodes. Loops are more likely to appear where nodes are in a higher density. We further repeat our experiments with an active duty-cycle of 75%, 85% and 95%, and concurrently deliver all the packets between the 100 source-destination pairs. If a loop appears, its starting time and avoiding time are recorded. Figure 5.9 shows the number of loops at each time slot. It tells us that a loop appears very quickly in such kinds of active duty-cycle. The earliest loop starts at the second time slot and the latest loop is detected at time slot 45.

5.5.3 Overheads

Our algorithm assists GPSR to avoid loops during packet delivery with the cost of storing extra information. In GPSR, a packet only needs to record the followings: source, destination and the place of entering current face. With our algorithm, all the regular paths within the current face are recorded as well. We record memory occupied by recording regular paths for each packet whenever a loop is detected. Network size 600×200 in Figure 5.10 shows the minimal, maximal and average memory usage of running 100 source destination pairs in *greenOrbs* topology with 80% active duty-cycle. At most 14 visited nodes are required to be stored at a packet. Minimal memory occupancy is quite close to average memory occupancy. It is because most loops are within a few nodes as illustrated by Figure 5.8.

We further scale up and down the network size without changing the node density. Four networks with size 300×100 , 450×150 , 750×250 and 900×300 are simulated.

Each of them runs with 100 source-destination pairs under 80% active duty-cycle. The corresponding results are shown in Figure 5.10. There is a slight increase at the maximal memory occupancy while network scales up. However, minimal and average memory occupancy are quite stable when the network scales up because most of the loops are within a few nodes.

With our algorithm, GPSR records the information of 5 extra nodes in each packet on an average. In *greenOrbs*, the size of a packet is 50 Byte. 9 bits are enough to represent any of the 444 different IDs. That is to say, there is a 11% overheads on an average.

5.6 Summary

In this chapter, we observed that existing face routing algorithms are tailored for sensor networks with uniform communication radius and suffer packet transmission failures when sensor nodes are with nonuniform communication radius. To thoroughly solve such a problem, we analyzed the cause of delivery failures. Based on our analysis, we proposed a simple algorithm to improve face routing performance by avoiding loops during packet delivery. Experiments were conducted to verify our algorithm. The results confirmed the correctness of the proposed algorithm.

5.3.a: Before v_9 wakes up

5.3.b: After v_9 wakes up

Figure 5.3: Delivery failure caused by face split and merge

5.4.a: Before v_{14} sleep

5.4.b: after v_{14} sleep

Figure 5.4: Delivery failure caused by face merge and split

 5.5.a: F'_k is inside of F_k

 5.5.b: F'_k is on the right side of F_k without overlapping

 5.5.c: F'_k overlaps right side of F_k

 5.5.d: F'_k overlaps left side of F_k

 5.5.e: F_k is inside of F'_k

 5.5.f: F'_k is on the left side of F_k without overlapping

Figure 5.5: Possible new major face positions due to topology changes

5.6.a: Packet routing path in face merge&split with our algorithm

5.6.b: Packet routing path in Case 2 with our algorithm

Figure 5.6: Successful packet delivery of GPSR with our algorithm

Figure 5.7: Comparison between GPSR solely apply and with our algorithm in success rate. N is the number of source-destination pairs has been executed

Figure 5.8: Loops recorded under 80% active duty-cycle with 100 source-destination pairs. Arrows indicate packet delivery direction.

Figure 5.9: The number of loops at each time slot

Figure 5.10: Memory occupancy with different network size

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Energy efficiency is always an important issue in wireless sensor network applications. In this thesis, we addressed this issue through routing protocols. Two categories of routing protocols have been studied in this thesis, spatial index based routing and geographical routing.

Spatial index based routing is the major approach to support spatial query in wireless sensor networks. Many spatial index based routing varieties have been proposed for the support of spatial query. Existing works focus on optimizing a hierarchical structure for a sensor node parent-child relationship. Our work, on the contrary, is to improve the network lifespan by avoiding hierarchical structure rebuilding as much as possible.

We first introduced the multi-parent relationship to spatial index routing. Existing works only considered one to many parent-child relationship. By storing redundant parents' information, we extend the parent-child relationship to many-to-many. In addition, we provided parents selection mechanism to balance the energy usage of candidate parents.

We move our scenario to high density wireless sensor networks where spatial index routing is inefficient hereafter. By introducing the idea of virtual node, we are able to

run spatial index in high density sensor networks and dramatically prolong network lifespan.

Spatial index routing is anyhow a centralized solution. It is not scalable in large scale wireless sensor networks. Therefore, we move our focus on a stateless solution, namely, geographic routing.

Most of geographic routing variants adopt face routing to guarantee delivery and they are all designed for static wireless sensor networks. Simply apply existing geographic routing algorithms in duty cycling wireless sensor networks may not guarantee delivery.

We observed delivery failures in duty cycling wireless sensor network and carefully studied the reasons. Based on our analysis, we proposed a geographic routing algorithm to maximally guarantee delivery in duty cycling wireless sensor networks.

6.2 Future Work

Delivery guarantee of most geographic routing algorithms is based on the assumption that face routing is executed on a planar graph, which means a wireless sensor network should be planarized in distributed manner. Based on the assumption that each sensor node has exactly the same communication radius, a wireless sensor network can be planarized to support the correctness of face routing. However, this requirement cannot be verified in a real network. Even if they are equipped with the same antenna, factors such as terrain, working condition, obstacles and remaining energy can change their communication radius. In our future work, we will study the performance of geographic routing algorithms in a sensor network where each sensor node has different communication radius and analyze how delivery guarantee can be achieved under such condition.

In addition, some preliminary works on sensor network energy usage prediction and privacy preserving have been done. We will further explore these two areas in our future work.

Markov Model has proved its feasibility of predicting the energy state of sensor nodes. Thus, users can monitor the energy state of sensor nodes in real-time without querying them frequently. However, a stationary state transition probability is required to apply the Markov Model, which means the prediction is only applicable to schedule-driven sensor networks rather than trigger-driven sensor networks. In our preliminary work, we studied to use the Markov Model to make predictions in trigger-driven sensor networks. By considering event distributions and query patterns, our proposed method managed to predict sensor node energy level information of trigger-driven sensor networks. As part of our future work, we will explore sensor network with dynamical state interval, such as demand-based state interval. Besides, we will also explore other prediction methods other than the Markov Chain.

Ubiquitous deployment of wireless sensor networks provides great convenience for environment monitoring. However, it also brings the risk of violating privacy. Sensitive sensor data disclosed to malicious parties may cause unexpected loss. In our preliminary work, we proposed a privacy-preserving in-network aggregation protocol for wireless sensor network based on the concept of data slicing, mixing and merging with a novel shared key management scheme. Our protocol allows performing in-network aggregation in a sensor network while keeping the privacy of participants. Although we only studied additive aggregation in this work, our protocol can be easily extended to other aggregation functions, including average, count and many other functions based on aggregation as long as they can be reduced to additive aggregation function. In the future work, we are going to extend our work to other aggregation methods, such as *Min/Max*. We are also going to study other mutual authentication algorithms with lower computation and communication overheads.

Author's Publications

- W. Zha, W.K. Ng and D.H. Tran, “Energy-efficient return-path routing in wireless sensor networks”, *TENCON, IEEE Region 10 Conference*, Pages 1-6, 2009, Singapore
- W. Zha, W.K. Ng, “Predict Energy Consumption of Trigger-Driven Sensor Network by Markov Chains”, *ICDCS Workshop*, Pages 350-356, 2011, Minneapolis, U.S.A
- W. Zha, W.K. Ng, “Privacy-preserving In-network Aggregation in Wireless Sensor Network”, *SENSORNETS*, Pages 223-228, 2012, Italy
- W. Zha, W.K. Ng, “Moving Geographic Routing to Practice”, *IEEE Cyber*, Pages 144-147 2012, Thailand
- W. Zha, W.K. Ng, “Geographic Routing with Cross Links”, *KICSS*, TBA 2012, Australia
- W. Zha, W.K. Ng, “Prolonging Lifespan of Sensor Networks using Redundant Nodes”, *Journal of Ad Hoc Sensor Wireless Networks* 19(1-2):1-19, 2013
- D.H. Tran, W.K. Ng and W. Zha, “CRYPPAR: An efficient framework for privacy preserving association rule mining over vertically partitioned data”, *ICICS*, Pages 1-5, 2011, Singapore

- D.H. Tran, H.L. Nguyen, W. Zha and W.K. Ng, “Towards security in sharing data on cloud-based social networks”, *TENCON, IEEE Region 10 Conference*, Pages 1-6, 2009, Singapore

References

- [1] C4isr journal. In *<http://www.c4isrjournal.com/>*.
- [2] Wind and structural health monitoring system. In *<http://en.allexperts.com/>*.
- [3] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *Proceedings of the ACM international conference on Management of data*, volume 19, pages 322–331, Atlantic City, NJ, USA, 1990.
- [4] E. S. Biagioni and K. W. Bridges. The application of remote sensor technology to assist the recovery of rare and endangered species. In *International Journal of High Performance Computing Applications*, volume 16, pages 315–324, 2002.
- [5] Edoardo Biagioni and Galen Sasaki. Wireless sensor placement for reliable and efficient data collection. In *Proceedings of the 36 th Annual Hawaii International Conference on System Sciences*, 2003.
- [6] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *journal of Wireless Netowrks*, pages 609–616, 2001.
- [7] David Braginsky and Deborah Estrin. Rumor routing algorithm for sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on*

REFERENCES

- Wireless sensor networks and applications*, pages 22–31, Atlanta, Georgia, USA, 2002.
- [8] B.G. Celler, W. Earnshaw, E. Ilsar, L. Betbeder-Matibetb, M. F. Harrisb, R. Clarkc, T. Hesketha, and N. H. Lovelld. Remote monitoring of health status of the elderly at home. a multidisciplinary project on aging at the university of new south wales”. In *International Journal of Bio-Medical Computing*, pages 147–155, Baltimore, MD, USA, 1995.
- [9] B.G. Celler, T. Hesketh, W. Earnshaw, and E. Ilsar. An instrumentation system for the remote monitoring of changes in functional health status of the elderly at home. In *Proceedings of the 16th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 908–909, Baltimore, MD, USA, 1994.
- [10] Alberto Cerpa, Jeremy Elson, Michael Hamilton, Jerry Zhao, Deborah Estrin, and Lewis Girod. Habitat monitoring: application driver for wireless communications technology. In *SIGCOMM LA '01: Workshop on Data communication in Latin America and the Caribbean*, pages 20–41, San Jose, Costa Rica, 2001.
- [11] Coskun Cetinkaya and Edward Knightly. Opportunistic traffic scheduling over multiple network paths. In *IEEE International Conference on Computer Communications*, pages 1928–1937, 2004.
- [12] Silvio Croce, Francesco Marcelloni, and Massimo Vecchio. Reducing power consumption in wireless sensor networks using a novel approach to data aggregation. *The Computer Journal*, 51:227–239, 2008.
- [13] Thanh Dang, Sergey Frolov, Nirupama Bulusu¹, Wuchi Feng, and Antnio Baptista. (Corie) observation network for data assimilation using genetic algorithms.

REFERENCES

- In *3rd IEEE International Conference on Distributed Computing in Sensor Systems*, pages 253–266, Santa Fe, New Mexico, USA, 2007.
- [14] Murat Demirbas and Hakan Ferhatosmanoglu. Peer-to-peer spatial queries in sensor networks. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, pages 32–39, Linkoping, Sweden, 2003.
- [15] Raphael Finkel and Jon Bentley. Quad trees: A data structure for retrieval on composite keys. In *Acta Informatica*, pages 1–9, 1974.
- [16] H. Frey and I. Stojmenović. Geographic and energy-aware routing in sensor networks. In *Handbook of Sensor Networks*, pages 381–415, 2005.
- [17] H. Frey and I. Stojmenovic. On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 390–401, 2006.
- [18] S Ganeriwal, R Kumar, and M B Srivastava. Timing-sync protocol for sensor networks. In *International conference on Embedded networked sensor systems*, pages 138–149, 2003.
- [19] Yong Gao, Kui Wu, and Fulu Li. Analysis on the redundancy of wireless sensor networks. In *International Workshop on Wireless Sensor Networks and Applications*, pages 108–114, San Diego, CA, USA, 2003.
- [20] S. Giordano, I. Stojmenovic, and L. Blazevic. Position based routing algorithms for ad hoc networks: A taxonomy. In *Ad hoc wireless networking*, pages 103–136, 2004.

REFERENCES

- [21] Shuo Guo, Yu Gu, Bo Jiang, and Tian He. Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, pages 133–144, 2009.
- [22] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the ACM international conference on Management of data*, pages 47–57, Boston, Massachusetts, USA, 1984.
- [23] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *International conference on system sciences*, pages 3005–3014, Hawaii, USA, 2000.
- [24] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari 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*, pages 174–185, Seattle, Washington, United States, 1999.
- [25] Ting-Chao Hou and Victor Li. Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications*, pages 38–44, 1986.
- [26] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of ACM MobiCom Conference*, pages 56–67, Boston, USA,, 2000.
- [27] W. Kanarkard, W. Taweeprawadej, R.G. Adams, and N. Davey. Flood and draught automated local evaluation in real time system for mahasarakham

REFERENCES

- province. In *the International MultiConference of Engineers and Computer Scientists 2009*, Hong Kong, 2009.
- [28] Brad Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, 2000.
- [29] Abtin Keshavarzian, Huang Lee, and Lakshmi Venkatraman. Wakeup scheduling in wireless sensor networks. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 322–333, 2006.
- [30] Leonard Kleinrock and John Silvester. Optimum transmission radii for packet radio networks or why six is a magic number. In *National Telecommunications Conference*, pages 4.3.1–4.3.5, Birmingham, Alabama, UK, 1978.
- [31] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *Proceedings 11th Canadian Conference on Computational Geometry*, pages 51–54, 1999.
- [32] Bhaskar Krishnamachari, Deborah Estrin, and Stephen B. Wicker. The impact of data aggregation in wireless sensor networks. In *International Conference on Distributed Computing Systems*, pages 575–578, Vienna, Austria, 2002.
- [33] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: of theory and practice. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 63–72, 2003.
- [34] Ben Leong, Sayan Mitra, and Barbara Liskov. Path vector face routing: Geographic routing with local face information. In *Proceedings of the 13TH IEEE International Conference on Network Protocols*, pages 147–158, 2005.

REFERENCES

- [35] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. In *Ambient Intelligence*, pages 115–148, 2005.
- [36] S. Lindsey and C. S. Raghavendra. Pegasus: Power-efficient gathering in sensor information systems. In *Aerospace Conference Proceedings, 2002. IEEE*, pages 1125–1130, Montana, USA, 2002.
- [37] Guikai Liu, Chunli Shan, Wei Gang, and Hongjiang Wang. Subarea tree routing in multi-hop wireless ad hoc networks. In *International Conference on Computational Science*, pages 1695–1699, Singapore, 2008.
- [38] Yunhao Liu, Yuan He, Mo Li, Jiliang Wang, Kebin Liu, Lufeng Mo, Wei Dong, Zheng Yang, Min Xi, Jizhong Zhao, and Xiang-Yang Li. Does wireless sensor network scale? a measurement study on greenorbs. In *INFOCOM, 2011 Proceedings IEEE*, pages 873–881, 2011.
- [39] Demirbas M and Xuming Lu. Distributed quad-tree for spatial querying in wireless sensor networks. In *IEEE International Conference on Communications*, pages 3325–3332, 2007.
- [40] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, Atlanta, Georgia, USA, 2002.
- [41] David Malan, Thaddeus Fulford-jones, Matt Welsh, and Steve Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *International Workshop on Wearable and Implantable Body Sensor Networks*, Imperial College London, United Kingdom, 2004.

REFERENCES

- [42] Curt Schurgers Mani and Mani B. Srivastava. Energy efficient routing in wireless sensor networks. In *MILCOM Proceedings on Communications for Network-Centric Operations: Creating the Information Force*, pages 357–361, McLean, USA, 2001.
- [43] Srivastava Mani, Muntz Richard, and Potkonjak Miodrag. Smart kindergarten: sensor-based wireless networks for smart developmental problem-solving environments. In *MobiCom: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 132–138, Rome, Italy, 2001.
- [44] Miklos Maroti, Branislav Kusy, Gyula Simon, and Akos Ledecz. The flooding time synchronization protocol. In *International conference on Embedded networked sensor systems*, pages 39–49, 2004.
- [45] Matthew J. Miller, Cigdem Sengul, and Indranil Gupta. Exploring the energy-latency trade-off for broadcasts in energy-saving sensor networks. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pages 17–26, 2005.
- [46] Zeeshan Hameed Mir and Young-Bae Ko. A quadtree-based hierarchical data dissemination for mobile sensor networks. In *Telecommunication Systems*, pages 117–128, 2007.
- [47] Kyungseo Park, Byoungyong Lee, and Elmasri. Energy efficient spatial query processing in wireless sensor networks. In *Advanced Information Networking and Applications Workshops*, pages 719–724, Canada, 2007.
- [48] Volkan Rodoplu and Teresa H. Meng. Minimum energy mobile wireless networks. In *IEEE Journal on Selected Areas in Communications*, pages 1333–1344, 1998.

REFERENCES

- [49] A. Rytter. Vibration based inspection of civil engineering structures. In *Ph.d disertation, department of building technology and structural engineering, Aalborg University*, 1993.
- [50] Christine Salamacha, Sterling Smoot, and Kathleen Farris. C4isrt in an operational context. In *JOHNS HOPKINS APL TECHNICAL DIGEST*, volume 21, 2000.
- [51] Timos K. Sellis, Nick Roussopoulos, and Christos Faloutsos. The R+-tree: A dynamic index for multi-dimensional objects. In *Proceedings of the 13th International Conference on Very Large Databases*, pages 507–518, San Francisco, CA, USA, 1987.
- [52] Amir Soheili, Vana Kalogeraki, and Dimitrios Gunopulos. Spatial queries in sensor networks. In *Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 61–70, Bremen, Germany, 2005.
- [53] David C. Steere, Antonio Baptista, Dylan McNamee, Calton Pu, and Jonathan Walpole. Research challenges in environmental observation and forecasting systems. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 292–299, Boston, Massachusetts, United States, 2000.
- [54] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 171–180, 2003.
- [55] Hanbiao Wang, Jeremy Elson, Lewis Girod, Deborah Estrin, and Kung Yao. Target classification and localization in habitat monitoring. In *ICASSP*, pages IV– 844–7 vol.4, 2003.

REFERENCES

- [56] Hanbiao Wang, Deborah Estrin, and Lewis Girod. Preprocessing in a tiered sensor network for habitat monitoring. In *EURASIP J. Appl. Signal Process.*, volume 2003, pages 392–401, 2003.
- [57] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *ACM MOBICOM*, pages 70–84, Rome, Italy, 2001.
- [58] Feng Xue and P. R. Kumar. The number of neighbors needed for connectivity of wireless networks. *Journal of Wireless Networks*, 10(2):169–181, 2004.
- [59] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. In *SIGMOD Rec.*, volume 31, pages 9–18, 2002.
- [60] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated, adaptive sleeping for wireless sensor networks. pages 493–506, 2003.
- [61] Yan Yu, Ramesh Govindan, and Deborah Estrin. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. 2001.
- [62] Yang Yu, B. Krishnamachari, and V. K. Prasanna. Energy-latency tradeoffs for data gathering in wireless sensor networks. In *IEEE International Conference on Computer Communications*, 2004.
- [63] Wei Zha, Sang-Hun Eo, Byeong-Seob You, Dong-Wook Lee, and Hae-Young Bae. Omsi-tree: Power-awareness query processing over sensor networks by removing overlapping regions. In *The Joint International Conferences on Asia-Pacific Web Conference (APWeb) and Web-Age Information Management*, pages 204–210, Huang Shan, China, 2007.

REFERENCES

- [64] Wei Zha, Wee Keong Ng, and Duc H. Tran. Energy-efficient return-path routing in wireless sensor networks. In *TENCON, IEEE Region 10 Conference*, pages 1–6, Singapore, 2009.