Coverage-oriented Network Scheduling and Location-directed Data Collection: Towards Energy-efficient Wireless Sensor Networks

ZHOU, Yangfan

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong

September 2009

Thesis Assessment Committee

Professor Man Hon Wong (Chairman)

Professor Michael R. Lyu (Thesis Supervisor)

Professor Kam-Wing Ng (Committee Member)

Professor Jiannong Cao (External Examiner)

Abstract of thesis entitled:

Coverage-oriented Network Scheduling and Location-directed Data Collection: Towards Energy-efficient Wireless Sensor Networks

Submitted by ZHOU, Yangfan
for the degree of Doctor of Philosophy
at The Chinese University of Hong Kong in September 2009

This thesis investigates various techniques to achieve energy-efficient Wireless Sensor Networks (WSNs) applied in environmental event detection and data collection.

We focus on two coverage-oriented problems for event monitoring WSNs. First, we study how to divide sensor nodes into a maximum number of disjoint subsets and each subset can maintain a required level of event coverage. This problem is important in prolonging the network lifetime since each subset can hence be scheduled to work in turn. We address this problem

with MAXINE (MAXimizing-ι Node-redundancy Exploiting). By maximizing ι, a fan-out index that measures the normalized minimum distance between the nodes, MAXINE well reduces the coverage redundancy of each subset so as to maximize the subset number. Second, we study how to schedule sensor nodes to enable downtime-free migration. We show that the system downtime, caused by system migration tasks such as software reprogramming, can be effectively eliminated by partitioning the sensor nodes into subsets and scheduling them to perform migration tasks successively with the rest still performing normal services. Several algorithms are then presented to attack such a partitioning problem.

For data collection WSNs, we provide a waypoint-based Geographic Data Reporting Protocol (GDRP), a light-weighted location-directed data forwarding scheme. GDRP adopts an intelligent strategy to select a best set of waypoints via which packets can well circumvent holes and barriers. It can thus reduce the energy of data forwarding.

Besides data forwarding, we also investigate contour mapping, an important technique to in-network-abstract the information of a monitored field. We design an energy-efficient On-demand Active Contour Service (OACS). OACS regresses the

field intensity function with kernel Support Vector Regression based on the sensor readings and the locations of the sensor nodes. Through an active and progressive learning algorithm, OACS determines the best set of nodes that should be turned on for obtaining sensor readings. It can hence accommodate a wide range of contour line/map precision requirements adaptively.

We verify our proposed protocols and algorithms with extensive simulation-based experiments, and the results confirm their advantages in achieving energy-efficient WSNs.

學位論文摘要

學位論文題目：
面向網路覆蓋的節點調度優化與基於位置信息的節點資料獲取：無線傳感器網路中的若干節能問題研究
提交人：周揚帆
學位：哲學博士

香港中文大學，二零零九年九月

本文研究無線傳感器網路在事件監測和數據收集應用中的若干節能問題。

對於無線傳感器網路在事件監測的應用，本文研究兩個面向網路覆蓋的節點調度優化問題。第一，我們研究如何把網路節點分成若干組，同時每組節點都能獨立保證所需的網路覆蓋指標。根據這個分組結果，我們可以做節點調度，使得每組節點可單獨工作而其他的組都可以進入節能狀態。很明顯，組數越多，網路可持續工作的時間越長。因此，我們提出一個稱爲 MAXINE 的算法，此算法通過優化一個度量點的分散性的指標來減少每組的節點冗餘，并據此來優化組數。第二，我們研究如何調度節點來實現在軟件升級過程中無線傳感器網路還能提供無間歇事件監測服務。我們指出可以通過分組升級的方式來避免軟件升級導致的系統暫停工作。具體做法是通過將網路分成若干組，讓每組依次做軟件升級，而其他各組還維持必需的事件監測服務。爲此，我們提出幾個演算法來實現這樣一個分組調度問題。

對於無線傳感器網路在数据收集的應用，我們設計了一個稱爲 GDRP 的用節點位置做導向的数据傳輸協定。GDRP 用一種智能的路點算法來規避網路中的障礙和空洞，從而縮短傳輸路徑達到節能的效果。 另外，除了数据收集，我們還研究了如何用等值線圖演算法來實現在網路內對傳感器数据的壓縮，我們設計了一個叫 OACS 的等值線圖算法，OACS 根據節點的讀數和位置用支持向量機做回歸來得到等值線圖。根據不同的用戶輸入的等值線圖的精度要求，OACS 通過一個主動節點選擇算法優化出一組需要工作的傳感器節點，而實現節能的效果。

大量的仿真實驗驗證了以上的協議和算法。其結果表明，這些協議和算法有效達到節省無線傳感器節點電池損耗的設計目標。

Acknowledgement

First of all, I owe my sincere gratitude to my supervisors, Prof. Michael R. Lyu and Prof. Jiangchuan Liu (who is with Simon Fraser University at Canada). Their support, patience, and encouragement are essential to my Ph.D. study. They are always my mentors guiding me on my long and winding road in pursuing an academic career. They have not only shaped the work described in this thesis, but also shaped the way I do my research.

I appreciate Prof. Kam-Wing Ng and Prof. Man Hon Wong at The Chinese University of Hong Kong, and Prof. Jiannong Cao at The Hong Kong Polytechnic University for their precious time to serve as my thesis assessment committee members during my Ph.D. study. Their warm comments and constructive suggestions have helped a lot in improving this thesis.

I am happy that I had the chances to work with Dr. Guoliang Xing at Michigan State University and Dr. Dan Wang at

The Hong Kong Polytechnic University during my Ph.D. study. I warmly thank them for providing me many insightful suggestions on my research on wireless sensor networks. Working with them is always a pleasure.

I wish to thank my peer students in Prof. Michael R. Lyu's research group in Hong Kong and Prof. Jiangchuan Liu's research group in Canada. Discussing with them is always a pleasant experience. Especially, I wish to thank Dr. Xinyu Chen for working closely with me in many practical issues in wireless sensor networks, Ms. Junjie Xiong for providing me a lot of helps in the simulation studies in Chapter 4 of this thesis, Mr. Wujie Zheng for giving me many critical comments that help improve the work described in Chapters 3 and 5 of this thesis, Mr. Feng Wang for sharing with me his research experiences in wireless sensor networks, and Dr. Haixuan Yang for teaching me a lot in machine learning.

I am also very thankful to my best friends in Hong Kong over these years, Xinyu and Sun Jin, Zhang Kun and Zhang Wan, Erica and Jhan, Li Gang, and Steven for the warm supports they have provided and the good time we share together.

Most of all, I feel very grateful to my mother Ms. Feng Lin, my father Mr. Xingquan Zhou, my mother-in-law Ms. Fenggen

Tao, my father-in-law Mr. Longbao Xue, my sister Yinghong, and my wife Hong Xue. In my life, they are always the endless source of warm support and love. Without them, every single achievement of mine would be meaningless. Especially, my gratitude to Hong is beyond words. Over all these years, her understanding, her patience, her suggestions, her encouragement, and most of all, her love are substantial to my research work. Without her, my Ph.D. study would be impossible. To her, this thesis is dedicated.

This work is dedicated to my wife Hong Xue.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background Study

1.1 Wireless Sensor Networks

Recent advances in sensors, wireless technologies, and microcontrollers have enabled the integration of these components into one tiny battery-powered device [5, 49, 76, 83]. Such tiny devices, often referred to as *wireless-integrated sensor nodes*, or in short, *sensor nodes*, can be in-situ deployed to detect certain environmental events or collect specific environmental data with their physical sensors. The sensor nodes are capable of forming a wireless sensor network (WSN) with a bunch of layered standardized protocols, via which the sensor nodes can process the environmental data collaboratively, and convey the data to a data sink with multi-hop wireless communications [3, 21, 30].

In-situ sensing with WSNs has long been advocated as a vi-

able method for environmental event detection and data collection due to their low costs in implementation, deployment, and maintenance [71, 72]. This section provides a systematic overview of the in-situ sensing systems with WSNs. We will first examine the hardware implementations of a typical wireless-integrated sensor node. Then, we will demonstrate the architecture of a general WSN, which is followed by a summary of the major characteristics of WSNs.

1.1.1 Wireless-integrated Sensor Nodes

Early implementations of sensor nodes include the μAMPS sensor nodes [76], the WINS sensor nodes [5, 83], the XSM sensor nodes [28, 33], and the MICA Mote sensor nodes [49]. The hardware architectures of these implementations are generally the same, which are followed by most of the current sensor node implementations. Figure 1.1 overviews the typical hardware architecture adopted by most of the state-of-the-art low-end sensor nodes (*e.g.*, IRIS Mote [20]). It contains the following components:

- A Microcontroller Unit (MCU). In general, the MCU adopts the Harvard architecture, where the storage and signal pathways for instructions and data are separated. The MCU

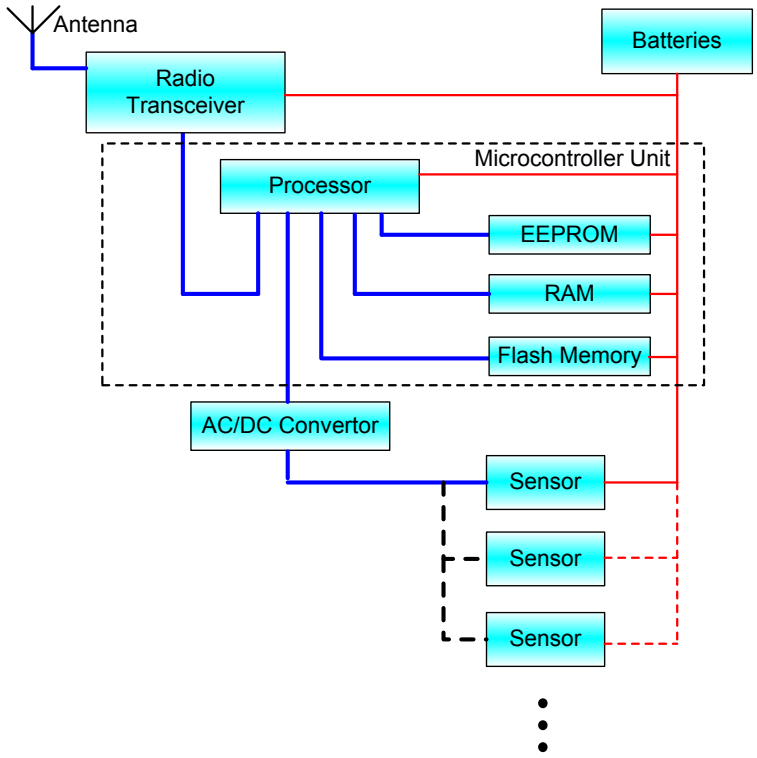


Figure 1.1: Typical hardware architecture of sensor node

processes the sensor data, controls and monitors the behaviors of the sensor node, and runs the required software such as the networking protocols. The MCU contains what follows:

- A processor, which takes charge of the node's control and computation operations. Its working frequency ranges from several to tens of MHz.
- An EEPROM, which saves the program codes. Its capacity is several KBytes.

  - A RAM, which saves the program run-time data. Its capacity is several KBytes.
  - A flash memory, which can be flexibly used to cache the sensor data or save the program codes. It has a relatively larger capacity, *e.g.*, 128 KBytes for ATmega128, an MCU largely adopted by the typical low-end sensor nodes [20].

- One or more physical sensors. Examples of sensors include thermoelectrical sensors, photosensitive sensors, barometric sensors, and humidity sensors to measure the environments; acoustic sensors to capture the sounds of interest; infrared sensors to capture the existence of the objects of interest, and seismic sensors to measure earth vibrations. The types of sensors employed are determined by the applications of the in-situ sensing tasks.

- An AC/DC convertor, which converts analog sensor outputs to digital signals so that it can be read and processed by the MCU.

- A radio transceiver, which is IEEE 802.15.4-compliant usually. It works on 2.4 GHz frequency and its communication range is tens of meters.

- The power supply unit, which consists of two AA batteries.

The size of a typical sensor node is comparable to a matchbox. Figure 1.2 shows the IRIS Mote, a typical commercial sensor node hardware platform [20].

Figure 1.2: IRIS Mote: A typical low-end sensor node hardware platform

1.1.2 Networking Sensor Nodes

Individual sensor nodes are with low sensing, processing, and communication capabilities due to their low-cost implementation and small size. Networking a large number of sensor nodes can enhance the amount, as well as the accuracy, of the information obtained by the sensor nodes [31, 32]. As a result, in most proposed implementations of in-situ environmental sensing with sensor nodes, a large number of sensor nodes are deployed to form an ad hoc multi-hop wireless network with the radio transceivers they are equipped. Such wireless networks are referred to as WSNs [32].

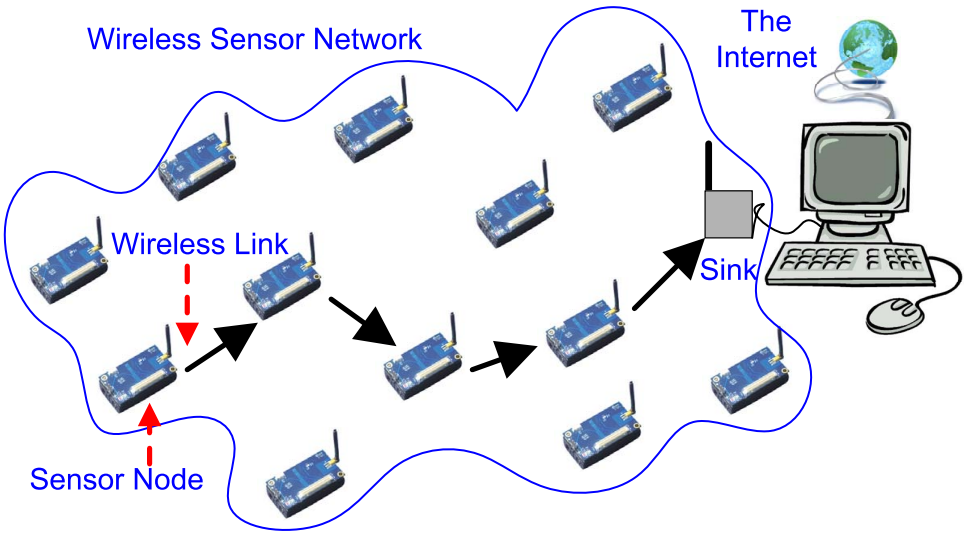


Figure 1.3: A representative WSN architecture

In WSNs, each sensor node supports a multi-hop routing algorithm, through which the collected data of each node can be conveyed to a data sink (as known as a base station). The data sinks are usually more powerful devices with abundant computational capabilities, *e.g.*, hand-held devices such as PDAs and laptop computers. They are capable of communicating with other computers or even connecting to the Internet. The sinks are where the outside world obtains the data from the sensor nodes and controls the behaviors of the sensor nodes. Figure 1.3 shows a representative WSN architecture [3].

1.1.3 Characteristics of WSNs: A Summary

WSNs are similar to Mobile Ad Hoc NETworks (MANETs) as they are both multi-hop wireless networks. But WSNs are very different from traditional data networks including MANETs. This section summarizes the major characteristics of WSNs.

Large-scale, unattended, and self-organizing networks

WSNs are usually with large scale and high node density. The number of sensor nodes in a network may be several hundred or even reach over a thousand. Such a large number of sensor nodes usually work in a human-unattended manner after they have been deployed. Moreover, in most WSN application scenarios, the physical locations of the sensor nodes cannot be predetermined. For example, the sensor nodes may be randomly dropped by an aircraft in a human-inaccessible territory. Hence, a WSN is required to function in a completely autonomous manner. The WSN protocols should be self-organizing, which can work without any manual configuration of each individual sensor node.

Resource-constraint sensor nodes

First, WSNs suffer from limited energy supply. Sensor nodes are powered by small irreplaceable batteries. They are usually working in an human-unattended manner, sometimes even in a human-inaccessible territory [70, 23]. It is impractical to revive a sensor node after its battery energy is drained. Hence, the protocols for WSNs should be energy-efficient so as to extend the network lifetime. Energy-efficiency must be a major design consideration of the WSN protocols. Second, WSNs suffer from low computational capacity. A sensor node has low memory capacity (usually several KBytes [20]) and its computational speed is very slow (usually by a CPU working in several MHz frequency[20]) due to its low-cost design, small size, and energy constraint. Simplicity is hence another design goal of the WSN protocols.

Application-specific networks

A WSN may only need to satisfy specific application requirements, which is quite different to traditional general-purpose networks. The protocol design in WSNs is not required to achieve compatibility to the existing protocols for traditional general-purpose networks. For example, unlike in MANETs, we

do not have to build up an IP addressing mechanism for WSNs. It is not necessary to confine the protocol stack design in WSNs to traditional TCP/IP layering thought. In other words, the protocols for WSNs can be totally new. With such flexibility, We can optimize the WSN protocols based on the characteristics of the applications running over the networks.

Location-aware networks

A global identity-based addressing mechanism, which is required by traditional data networks, is unnecessary or even impractical for WSNs. The applications of WSNs are mainly on environmental data collections and event detections. They would generally care more about the sensor readings of some particular locations or whether some specific events have occurred, rather than which node is currently reporting data. Hence, it is usually required that each in-network sensor node is aware of its geographic location as the location information is necessary to tag the sensor data or to identify/locate physical events. Location information is obtainable via some localization algorithms (*e.g.*, that proposed in [10, 114]).

In-network sensor data processing

Unlike in wired sensor networks where each sensor node is merely with sensing functionality, in WSNs each sensor node is equipped with an MCU. In-network data processing is hence possible in WSNs, which can be employed to reduce the volume of the data that need to be reported to the sink.

Stationary networks with one single, possibly mobile, sink

The network traffic feature of WSNs is different to MANETs where every node may require to communicate with all the others ones: traffic is usually in a peer-to-peer manner. In contrast, WSN traffic is usually in a many-to-one manner. Many nodes send data packets to one single sink. Moreover, the mobility feature of WSNs is different to MANETs. Unlike MANETs that contain mobile devices, WSNs contain stationary sensor nodes. But, the sink of a WSN may be a hand-held device such as a PDA or a laptop computer. It may be mobile. For example, it can be carried to the network area to collect the sensor data hourly, and each time its location may be different.

The above features of WSNs are the reasons that many network protocols for traditional data networks including MANETs are not suitable for WSNs. We have to investigate a bunch of

new network protocols and algorithms for data collection and network organization in WSNs which take the above network characteristics into account.

1.2 Applications of Wireless Sensor Networks

Since WSNs are application-specific networks, this section surveys the WSN applications to better understand the design considerations of WSN protocols.

Although the development of WSNs was originally motivated by military applications such as battlefield surveillance, a great variety of potential industrial and civilian WSN applications have been proposed in recent years [87]. Although most of the proposed applications of WSNs are only in experimental phase and they just serve as prototypes and test-beds to help researchers find out the challenges and verify the existing ideas, real-world applications of WSNs are expected to flourish and have a great impact in our life in a foreseeable future.

Example WSN applications include environmental monitoring [24, 81, 93, 113] and measurements [121], habitat monitoring [70], drink water quality monitoring [2, 25], industrial process monitoring [60, 61], structure health monitoring [58, 82, 117], intruder detection [4], and vehicle classification and tracking [27,

94]. Examples of other related applications include a biomedical application which aims at turning artificial retina into reality [92] and a "smart kindergarten" application which introduces WSNs into childhood education [100]. These applications can generally be classified into two categories based on their characteristics, namely, event detection and data collection.

In an event detection WSN, each sensor node is responsible for sensing the phenomena of interest around the node. The network then coordinates to perform specific event detection tasks, *e.g.*, whether a certain type of vehicles are passing by [27] or whether there is an intrusion event [4]. When an event of interest is successfully detected, the network will report an alarm to the sink. For such networks, maintaining the necessary quality of event detection (*i.e.*, coverage) in an energy-efficient manner is a basic requirement. Forwarding sensor data, on the other hand, is a light-weighted task since the traffic volume in such an application is generally low.

For data collection WSN applications, the major task of the WSNs is to stream the sensor data to a sink so that the sink can instantly obtain the environmental information of interest. In contrast to event-monitoring WSNs, data collection WSN applications generally need to continuously report data to the

sink. Hence, optimizing the data forwarding scheme is naturally a major concern of such applications to achieve energy efficiency. Furthermore, since the intensity of a physical field being monitored is continuous in nature, the readings of nearby sensor nodes are correlated. In-network data processing can be employed to reduce such redundancy, and hence to lighten the traffic from the sensor nodes to the sink, which can save the energy consumption of data collection.

1.3 Thesis Scope and Contributions

This thesis investigates two major problems, namely, coverage-oriented network scheduling and location-directed data collection, for event detection WSNs and data collection WSNs respectively, to achieve energy-efficiency in WSN in-situ sensing. This thesis is based on our publications in [128, 129, 130, 131, 135].

1.3.1 Coverage-oriented Network Scheduling

Since maintaining a necessary coverage quality is a basic requirement for event detection WSNs, this thesis studies two important coverage-oriented network scheduling problems. The first is how to divide a set of sensor nodes into a maximum

number of disjoint subsets and each subset can cover the entire network territory. Such a subset number actually indicates the network life time, since the subsets can then be scheduled to work in turn. We solve this problem with an effective geometric algorithm MAXINE (MAXimizing-ι Node-redundancy Exploiting). MAXINE is based on a fan-out index, namely ι, which measures the normalized minimum distance between the sensor nodes [126, 136]. We show that by maximizing ι, MAXINE well exploits the coverage redundancy of each subset so as to maximize the number of subsets. The performance study of MAXINE suggests that MAXINE achieves good coverage efficiency with much lower computational cost. More importantly, it can be easily extended to a distributed and localized implementation.

Second, we study how to schedule the sensor nodes to enable downtime-free migration. Current sensor platforms are usually equipped with reprogramming modules. System migration tasks such as software reprogramming, however, will interrupt normal sensing operations of a node and disable the network to detect critical events, posing a severe threat to many sensitive event detection applications. We show that such a system downtime caused by the migration tasks can be effectively eliminated, by partitioning the sensors into a given number of disjoint subsets

and scheduling them to perform the migration tasks successively with the rest still performing normal services. Several effective algorithms are then presented and we further extend our solution to a practical decentralized implementation. We demonstrate that our algorithms achieve good balance between the sensing quality and system migration time.

1.3.2 Location-directed Data Collection

For data collection WSNs, we study their two major tasks: data forwarding and in-network data processing.

Since WSNs are generally location-aware networks, the location information of the sensor nodes can be employed to design data forwarding protocols in WSNs. Location-directed data forwarding is favorable for WSNs due to its simplicity and low-overhead. However, WSNs are usually subject to complicated environmental factors. Network holes and barriers are inevitable in practical deploying environments. A location-directed data forwarding scheme should well tolerate holes and barriers energy-efficiently. We hence specifically tailor a waypoint-based Geographic Data Reporting Protocol (GDRP). As a location-directed forwarding scheme, GDRP is light-weighted and hence well suits WSNs. But unlike other approaches that often find suboptimal

paths, GDRP adopts an intelligent strategy to select a best set of waypoints via which packets can efficiently circumvent holes and barriers, and it can thus find better paths.

Besides data forwarding, how to process data in network is also a critical issue in data collection applications, since this can reduce the number of packets and thus save energy. As contour mapping is a general technique to abstract the information of a monitored field, we design an energy-efficient On-demand Active Contour Service (OACS) for WSNs. OACS regresses the field intensity function with kernel Support Vector Regression based on the sensor readings and locations of the sensor nodes. It can adaptively accommodate a wide range of contour line/map precision requirements: For applications of low precision, only a minimum set of nodes are scheduled in the on-duty mode while others are sleeping for conserving energy. For applications of high precision, through an active and progressive learning algorithm, OACS determines the best set of nodes that should be turned on for improving the contour line/map precision. Our performance evaluation based on diverse realistic models demonstrates that it can significantly conserve energy for various application requirements.

1.4 Thesis Organization

This thesis is organized as follows. In Chapter 2, we investigate how to divide a set of sensor nodes into a maximum number of disjoint subsets such that each subset can cover the entire network territory. Chapter 3 studies how to schedule the sensor nodes to enable downtime-free migration. Data forwarding is investigated in Chapter 4, where we propose GDRP. Chapter 5 presents OACS. Finally, we conclude this thesis in Chapter 6.

□ **End of chapter.**

Chapter 2

Coverage-oriented Network Partitioning in WSNs

Summary

This chapter addresses a critical coverage-oriented partitioning problem: How to divide a set of sensor nodes into a maximum number of disjoint subsets, so that each subset can cover the entire network territory. We presents a fan-out index ι for geographically-located points, which measures the normalized minimum distance between the points. Maximizing ι of the points makes their Delaunay triangulation graph be a lattice of equilateral triangles. Such a structure achieves the lowest redundancy of coverage if each point represents the center of a disc, and hence can be a good measure of coverage efficiency for event detection wireless sensor networks. With this fan-out index ι, we solve the coverage-oriented partitioning problem through an effective geometric algorithm MAXINE (MAXimizing-ι Node-redundancy Exploiting). We evaluate the performance of MAXINE through extensive simulations and compare it with existing algorithms. Our solution suggests that MAXINE achieves good coverage efficiency yet with much lower computational cost.

2.1 Overview

In WSNs, each on-duty node (*i.e.*, a node that is performing the sensing/communication task) is only responsible for sensing the physical phenomena within the geographic region around the node. With the individual sensor readings, the whole network can coordinate to perform certain event detection tasks in the entire network territory [32]. This motivates a new domain of *coverage-oriented* network management issues, where maintaining a certain degree of network coverage, *i.e.*, a necessary quality of the event detection, becomes the basic requirement. It has long been known that to maintain efficient coverage (*i.e.*, minimizing the number of the on-duty nodes needed for coverage maintenance), the on-duty nodes should be distributed as evenly as possible [7, 57]. In other words, the on-duty nodes should fan out well in the network territory. Unfortunately, it remains unclear how the fan-out of a set of nodes can be evaluated quantitatively.

We propose a fan-out index ι for a set of points [126, 136]. ι is the minimum distance between each pair of points normalized by their average distance [126, 136]. We find that maximizing ι results in an equilateral triangle lattice of points if the points are moveable. Such a structure gives the best coverage efficiency

if each point represents a sensor node [7, 57, 115]. Hence, ι of a set of nodes can serve as a good microscope to indicate the coverage redundancy of the nodes. This suggests that the idea of maximizing ι is applicable to the coverage-oriented problems in WSNs. We demonstrates it through attacking a coverage-oriented network partitioning problem, *i.e.*, how to divide the sensor nodes into disjoint subsets such that the subset number is maximized, while each individual subset can maintain network coverage.

The coverage-oriented network partitioning problem is important to WSN management. The low-cost implementations and the field-working environments (which are sometimes even hostile, *e.g.*, in battle-field applications) of sensor nodes make them subject to failures and permanent damages. As a result, a WSN often contains many redundant nodes for backup purpose to enhance fault tolerance. To prolong the network lifetime, the sensor nodes are scheduled so that only a subset of them is on duty to maintain an acceptable level of network coverage. Obviously, the number of the subsets determines the WSN lifetime, and hence should be maximized.

This problem has been addressed through graph-theoretical approaches in existing works (*e.g.*, [13, 14]). These solutions,

however, suffer from high computational complexities that do not suit WSNs well. In contrast, we address this problem in a geometric perspective by greedily maximizing the ι index of the nodes in each subset. Through extensive simulation studies with different sensor coverage models (both a Boolean model and a collaborative probabilistic model), we demonstrate that our approach achieves comparably coverage efficiency with much lower computational cost. Moreover, unlike the existing schemes, our approach is easy to be implemented in a distributed and localized manner.

The rest of this chapter is organized as follows. Section 2.2 briefly surveys the related work. Section 2.3 formulates the coverage-oriented network partition problem in WSNs. In Section 2.4, we introduce the fan-out index ι for a set of points. We then illustrate how we solve the coverage-oriented network partition problem with an algorithm based on the ι fan-out index in Section 2.5. Section 2.6 presents our comprehensive experimental studies. We finally provide the conclusion remarks in Section 2.7.

2.2 Related Work

Many approaches target on solving the coverage-oriented network partition problem. In [96], a network territory is divided into regions. The sensor nodes are grouped with the most-constrained least-constraining algorithm. It is a greedy algorithm in which the priority of selecting a given sensor is determined by how many uncovered regions this sensor covers and the redundancy caused by this sensor. Cardei *et al.* modeled the problem as one to find disjoint dominating sets [14]. The problem is known as NP-complete, and they proposed a graph-coloring based approximation. A similar problem of covering target points were studied in [13], which is again NP-complete and a mixed integer programming (MIP) approximation was proposed. Unfortunately, the size of such an MIP is usually very large and it is very time-consuming to solve it. Moreover, it is not easy to extend these centralized solutions to distributed and localized implementations that are expected by resource-constraint WSNs.

There are also many schemes proposed to schedule nodes in the sleeping/on-duty modes in an on-line manner: A sensor node determines its sleeping eligibility and the time it can sleep by querying its neighbors. Yan *et al.* introduced a differentiated

service in which a sensor node finds out its responsible on-duty duration with collaboration of its neighbors to ensure the coverage of sampled points [120]. Ye *et al.* developed the Probing Environment and Adaptive Sleeping (PEAS) algorithm where sensor nodes wake up randomly over time, probe their neighboring nodes, and decide whether they should start their surveillance work [122]. Wang *et al.* designed a Coverage Configuration Protocol (CCP) [111]. They suggested that the coverage degree of intersection-points of the neighboring nodes' sensing-perimeters indicates the coverage of a convex region. Xing *et al.* exploited a probabilistic distributed detection model with a protocol called Coordinating Grid (Co-Grid) [116].

Finally, there are many existing metrics to evaluate how a set of points distribute in a space. The *Mean Square Error* from these points to their mean location indicates how these points deviate from their central. In resource-sharing evaluation, the *Global Fairness Index* (GFI) is often employed to measure how even the resource distributes among these points [53], where the location of each point represents the amount of resource that belongs to the point. In WSNs, GFI can be used to calculate how even the remaining energy of sensor nodes is. These metrics, however, cannot show how well a set of points fan out in the

space.

2.3 Coverage-oriented Network Partitioning in Location-aware Wireless Sensor Networks

We consider a WSN deployed in a 2-dimensional area ϕ. There are totally n in-network sensor nodes denoted by $\{s_i\}_{i=1}^{n}$. Each node is responsible to monitor a circular area centering at the node with its radius equal to R. This circular area is called the *sensing region* of the node. Also consider that the network is location-aware: each sensor node s_i can know its approximate physical location $\mathcal{L}(s_i)$, which can be obtained by a GPS receiver or a localization algorithm (*e.g.*, that suggested in [10]). Location-awareness is crucial for an event detection WSN for locating the events of interests. It is a general assumption in existing works on coverage-oriented issues (*e.g.*, [13, 96]).

A WSN generally contains a large number of redundant nodes so as to achieve fault tolerance. In order to save energy and prolong the network lifetime, the sensor nodes can be divided into disjoint subsets. Each subset is able to maintain the required sensing tasks. The sensor nodes are scheduled according to the

subset they belong to. These subsets work successively: At any time, only one subset of sensor nodes are on-duty, while the rest are sleeping. Let $\mathcal{S}_j$ $(j = 1, 2, ..., m)$ denote the jth subset of the sensor nodes, where m is the total number of disjoint subsets. The coverage-oriented network partition problem is formulated as follows.

Problem 2.1: *The coverage-oriented network partitioning problem.*

Given:,

- A set of sensor nodes $\{s_i\}_{i=1}^{n}$ and the location $\mathcal{L}(s_i)$ of each sensor node.
- A *sensor coverage model* which quantitatively describes how a point P in ϕ is covered by the sensor nodes that are responsible to monitor this point. We call this quantity the *coverage* of P.
- A coverage requirement τ. When the coverage of a point is not smaller than this threshold, we say this point is *covered*.

Maximize: m, the number of the disjoint subsets.

Subject to:

- $\{s_i\}_{i=1}^{n} \supseteq \mathcal{S}_1 \cup \mathcal{S}_2 \cup ... \cup \mathcal{S}_m$
- $\mathcal{S}_j \cap \mathcal{S}_k = \varnothing$ ($\varnothing$ denotes an empty set); $\forall j, k$ and $j \neq k$

- ϕ can be covered by the sensor nodes in each subset $\mathcal{S}_j$ $(\forall j)$.

□

In other words, the coverage-oriented network partitioning problem is to address how to divide sensor nodes into as many disjoint subsets as possible, while each subset can maintain the coverage requirement of the entire network territory. Since the subsets can be scheduled to work successfully, a larger subset number implies a longer network lifetime.

For a complete description of the coverage-oriented network partition problem (*i.e.*, Problem 2.1), a concrete sensor coverage model should be given. Also we need a criterion to determine whether ϕ is covered. We discuss these two issues as follows.

2.3.1 Sensor Coverage Models

We consider two typical sensor coverage models, namely, the *Boolean coverage model* and the *collaborative probabilistic coverage model*.

The widely-adopted Boolean coverage model can largely capture the general characteristics of WSNs [111, 122]. In this model, it is assumed that a sensor node can always detect an event occurring in its responsible sensing region, *i.e.*, a circular area with radius R centering at the sensor [103, 116, 120]. Such

a model is similar to the disc-cover model in the geometry literature (*e.g.* [57]). Specifically, for a point with physical location L,

$$\mathcal{C}_j(L) = \begin{cases} 1, \text{ if } \exists s_i \in \mathcal{S}_j \ \& \ \|\mathcal{L}(s_i) - L\| < R; \\ 0, \text{ otherwise.} \end{cases} \tag{2.1}$$

where $\mathcal{C}_j(L)$ denotes the coverage of $\mathcal{S}_j$ at L in the network territory. In this model, the required coverage τ is equal to 1.

The collaborative probabilistic coverage model considers that a sensor node detects events based on the signal strength it senses. Event signals (*e.g.*, electromagnetic, acoustic, or photic signals) generally fade exponentially with the increasing of their transmit distance. The larger the distance, the weaker the event signals that can be sensed by the sensor, which reduces the likelihood of the event being detected. Based on this notion, we consider the probability $\mathcal{P}(L, s_i)$ that an event located at L can be detected by sensor s_i is:

$$\mathcal{P}(L, s_i) = \frac{\delta}{(\|\mathcal{L}(s_i) - L\|/\epsilon + 1)^{\beta}}, \tag{2.2}$$

where δ, β, and ϵ are constants related to the signal strength omitted by the event and how it fades. Since the sensor nodes conduct event-detection in a collaborative manner, an event can

be detected based on a cumulative probability. According to Equation (2.2), we have:

$$\mathcal{C}_j(L) = \begin{cases} 1 - \prod[1 - \frac{\delta}{(\|\mathcal{L}(s_k) - L\|/\epsilon + 1)^\beta}] \\ (\forall s_k \in \mathcal{S}_j \ \& \ \|\mathcal{L}(s_k) - L\| < R), \\ \text{if } \exists s_k \in \mathcal{S}_j \ \& \ \|\mathcal{L}(s_k) - L\| < R; \\ 0, \text{ otherwise.} \end{cases} \tag{2.3}$$

where $\mathcal{C}_j(L)$ denotes the coverage of $\mathcal{S}_j$ at location L in the network territory. This model is more general than that in [74, 66] which considers coverage as the *sum* of the sensed signal strengths of the sensors.

2.3.2 Coverage Sampling

Note that an algorithm in solving Problem 2.1 needs to guarantee to cover the entire network territory. To evaluate the coverage of a sensing field, the field can be sampled by some discrete points in the field. And then given a sensing model, *e.g.*, that described in Equation (2.1) or Equation (2.3), the coverage of these points can be quantified. The field is deemed covered if the coverage of each sampling point is not smaller than the requirement.

Sampling coverage with discrete points is in fact an approximation approach to ensure that the entire network territory is covered. Obviously, the larger the number of the sampling points, the better the approximation of the coverage sampling, which on the other hand also results in longer convergence time of the algorithm in solving Problem 2.1. Such tradeoffs are investigated in our experimental study presented in Section 2.6.

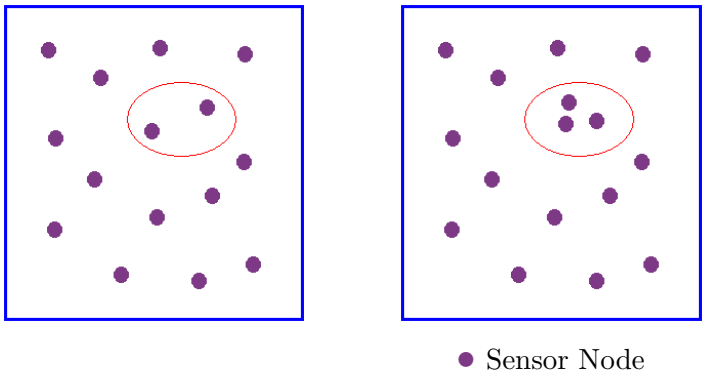
Besides the number of the sampling points, how to generate them is also an important issue. In this work, we sample coverage with four methods: 1) regular-lattice-based sampling, 2) random sampling, 3) quasi-random sampling, and 4) sensor-node-based sampling. Details on these sampling methods and their impacts will be presented in Section 2.6.

2.4 Normalized Minimum Distance ι: A Fan-out Index of Points

Under the constraint that the entire network territory is covered, it is desirable that the on-duty sensor nodes should be separated from each other as much as possible to achieve low redundancy. We need to quantize the quality of such separations. Consider two slightly-different WSN deployments in the

same territory demonstrated in Figure 2.1. Both can provide an acceptable level of network coverage. Network 2 in Figure 2.1(b) has one more node than Network 1 in Figure 2.1(a) in the elliptical region. The nodes in Network 1 fan out better than those in Network 2, since the nodes in Network 2 are closer to each other in the elliptical region. It is straightforward to see that Network 1 is better than Network 2 in terms of coverage efficiency, *i.e.*, Network 1 is with less node redundancy. This example shows that a small value of the minimum distance between the in-network nodes may indicate that the coverage efficiency of a network is bad somewhere in the network. The minimum distance between nodes may, to some extent, show how well the nodes fan out.

This consideration, however, is not complete. For example, if the minimum distance is $5m$, the coverage efficiencies are different for a network with 20 nodes and a network with 25 nodes that cover the same territory. We should normalize the minimum distance such that it can represent how well the nodes fan out for different network settings. We propose to normalize the minimum distance with the average distance between nodes. Such a normalized minimum distance, namely, ι, is formally defined as follows.



(a) (b)

Network 1 (a) and Network 2 (b) are two slightly different WSNs deployed in the same territory. The nodes in Network 2 are closer to each other in the elliptical region. As a result, the nodes in Network 1 fan out better than those in Network 2. It is straightforward to see that Network 1 is better than Network 2 in terms of coverage efficiency, for Network 1 has less node redundancy.

Figure 2.1: A demonstration of coverage efficiency

Suppose there are n points in a Euclidean space Ω with their coordinates being denoted by x_i $(i = 1, ..., n)$. Consider that n is larger than 2 and not all the points overlap (*i.e.*, we do not consider the case that $x_i = x_j$, $\forall$ i, j). ι is then calculated by [126, 136]:

$$\iota = \frac{\min(||x_i - x_j||)}{\mu} \qquad (\forall\ i, j; and\ i \neq j), \tag{2.4}$$

where $||x_i - x_j||$ is the Euclidean distance between points i and j, the $\min(\cdot)$ function calculates the minimum distance between each pair of points, and μ is the average distance between each

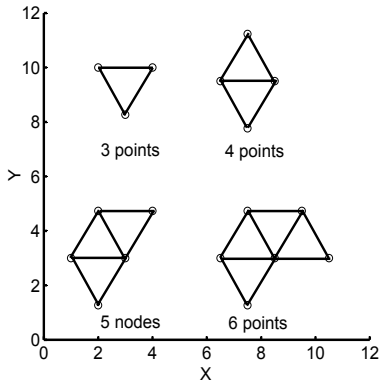


Figure 2.2: Resulting structures of 3, 4, 5, and 6 points when ι is maximized

pair of points:

$$\mu = \frac{(\sum_{\forall i} \sum_{\forall j \neq i} ||x_i - x_j||)}{n(n-1)}. \tag{2.5}$$

Obviously, ι is in interval $[0, 1]$. In a 2-dimensional space, it is equal to 1 if and only if n is equal to 3 and these three points form an equilateral triangle.

Consider that the location of each point x_i is a *variable*. How does the structure of these n points look like if ι is maximized? Given n =3, 4, 5, and 6, the graphs in Figure 2.2 are the resulting structures when ι is maximized in the 2-dimensional space. It is interesting to find out that the results are all equilateral triangle lattices. We have also calculated the structures for greater n's with a numerical algorithm where ι is maximized greedily. Again, we can observe that equilateral triangle lattices are al-

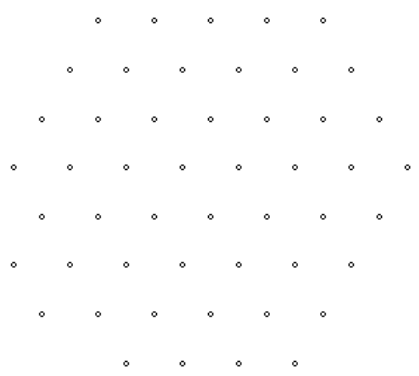


Figure 2.3: Resulting structure of 40 points when ι is maximized with a greedy algorithm

ways the results. Figure 2.3 shows a resulting structure when $n = 40$ as an example. We then draw the following thesis:

The ι-Thesis: *Given n points in a 2-dimensional space, ι of these points reaches the maximum value only if the Voronoi diagram*[1] *formed by these points is a net of equilateral hexagons,* i.e.*, these points forms a lattice of equilateral triangles.*

Suppose each point represents a sensor node and the sensor coverage model is the Boolean coverage model [103, 111, 116, 120, 122] with equal sensing radii of all nodes. It can be seen that such a network topology (*i.e.*, nodes with a honeycomb-like Voronoi diagram) results in lowest redundancy in coverage [7, 57]. In other words, a maximum value of ι of the nodes means the best fan-out of the nodes. Furthermore, ι is also sensitive

[1] A Voronoi diagram formed by a set of nodes partitions a space into a set of convex polygons such that points inside a polygon are closest to only one particular node [6].

to coverage redundancy. Two close nodes will result in small ι. This enlightens us that ι can be applied to solve coverage-oriented problems in WSNs. In our following discussions, we will demonstrate the effectiveness of employing ι in the coverage-oriented network partition problem formulated in Problem 2.1.

2.5 Maximizing-ι Node-redundancy Exploiting Algorithm for Coverage-oriented Network Partition

Since ι is a good microscope in indicating the coverage redundancy of a set of nodes, it is favorable that each subset in a solution for Problem 2.1 should have a large ι. Hence, the underlying thought of our Maximizing-ι Node-redundancy Exploiting Algorithm (MAXINE) algorithm is that a subset is formed by greedily maximizing its ι. In what follows, we first present a centralized implementation of MAXINE, and then describe how to extend it to be distributed and localized.

2.5.1 Centralized Implementation of MAXINE

Figure 2.4 illustrates the flow diagram of the centralized version of MAXINE. We call a node an *ungrouped node* if the node has

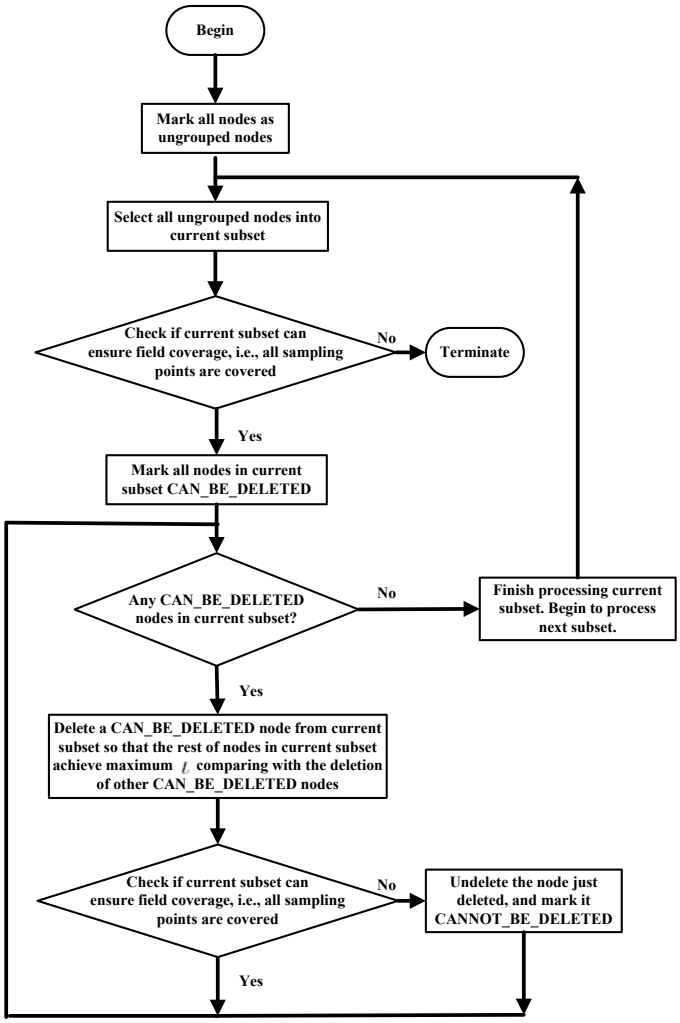


Figure 2.4: Flow diagram of MAXINE

not been grouped into any subset. Otherwise, we call the node a *grouped node*. Initially, all the nodes are ungrouped nodes. MAXINE tentatively selects all the ungrouped nodes into the current subset (initially, the subset is an empty subset $\varnothing$). And then, one by one, it removes nodes from the subset until the deletion of any node in the subset results in uncovered sampling

points. The selection criteria of the node to be removed are as follows.

- The deletion of the selected node does not result in any uncovered sampling points;
- The deletion of the selected node results in the maximum ι value of the current subset, comparing with the deletion of any other candidates (*i.e.*, those satisfying the first criterion) in the subset.

In this way, MAXINE first finds a subset that covers the entire network territory. It then continues to find the next subset with the same procedure, until the rest of all ungrouped nodes cannot guarantee to cover of all sampling points. It then stops with all the successfully-found subsets.

The centralized version of MAXINE runs at the base station of a WSN. It requires the location information of all in-network nodes. Such information is obtainable for location-aware WSNs. Specifically, the node locations may be sent to the base station along the shortest-path tree rooted at the base station[2] as follows. The leaf nodes in the tree send their locations first. When a non-leaf node collects all the location information of its

[2]Note that such a tree is a prerequisite for a WSN to report the sensing-data from sensor nodes to a base station, and hence can easily be utilized.

offspring nodes, it aggregates the locations of all its offspring nodes together with its own location in one packet and sends the packet to its parent node. Thus, every node needs to send only one packet to let the base station obtain all node locations.

Finally, when the base station finishes performing MAXINE, it can directly send the partitioning results to each in-network nodes if it is equipped with a powerful-enough antenna, or it can deliver the partitioning results in a multi-hop manner along the reverse direction of the shortest-path tree.

2.5.2 Distributed and Localized Implementation of the MAXINE Algorithm

In contrast to the centralized version that greedily maximizes the ι index of each resulting subset in an entire network scope, in the distributed and localized version, a node greedily maximizes ι of its neighborhood [126, 136]. We consider that a node s_i's neighborhood is the subnetwork consisting of s_i, its ungrouped *sensing neighbors* (*i.e.*, the neighbors with their distances to s_i within $2R$), and its grouped sensing neighbors that are in the same subset. Details are as follows[3].

Under the constraint that the coverage of its sensing region

[3]This part is based on my M.Phil. Thesis [126, 136].

should be larger than τ, the node removes the nodes in its neighborhood. The candidate to be removed satisfies that:

- It is an ungrouped node;
- The deletion of the node does not bring in any sampling point within the sensing region of s_i that cannot be covered by the neighborhood;
- The deletion of the node results in the largest ι of the neighborhood, compared with the deletion of any other candidates.

This node-removing procedure continues until no candidate can be found. Then all the ungrouped sensing neighbors that are not removed are grouped into the same subset of s_i. We call the sensing neighbors that are in the same subset of s_i the *in-subset sensing neighbors* of node s_i.

The distributed and localized MAXINE initiates the above procedure at a random-selected node. The node is grouped to the first subset. It calculates its in-subset sensing neighbors based on the procedure, and informs them the results. It then hands over to another in-subset sensing neighbor which have not performed such a procedure. The new node repeats the procedure until no such neighbors can be found. The first subset

that can cover the entire network territory is then formed.

After a subset is formed, another random-selected ungrouped node begins to group itself to a new subset and initiates the above procedure. MAXINE thus repeats to find subsets one by one. It stops when a node involving in this procedure finds that its sensing region cannot be covered by its neighborhood even if no node has been removed from the neighborhood.

2.6 Performance Study

2.6.1 Methods for Coverage Sampling

As the subsets of a solution for Problem 2.1 should guarantee to cover the entire network territory, the coverage is sampled with discrete points in the territory. The network territory is deemed covered if every sampling point is covered. We sample coverage with the following four methods.

Regular lattice (RL)

The network territory is divided into grids and the central point in each grid region is the sampling point for each grid.

Uniformly distributed random points (UDRP)

Sampling points are randomly selected in a uniform manner in the network territory.

Quasi-Random sequences (HTS and HMLS)

Quasi-random sequences which have low *discrepancy* (a measure of uniformity for the distribution of the points) have been widely employed in Quasi Monte Carlo methods [45, 46]. We consider two typical quasi-random sequences, Halton sequence [45] and Hammersley sequence [46]. Both sequences can achieve asymptotically optimal discrepancy and they are easy to be constructed. We linearly map the 2-dimensional Halton sequence and the 2-dimensional Hammersley sequence into network territory to generate the locations of the sampling points we need. We use the abbreviations HTS and HMLS to represent the Halton-sequence-based method and the Hammersley-sequence-based method, respectively.

Sensor-node-based points (SNB)

In this method, the sampling points are generated on-line, based on the sensor nodes in each subset. In the Boolean coverage model case, we evenly select the points on the border of each

node's sensing region (*i.e.*, the perimeter of the circle centering at the node with radius R). If such a point is in the territory being monitored by the network, it is then a sampling point. In the collaborative probabilistic model case, we evenly divide the sensing region of a node into several areas and regard the central point in each area as a sampling point if the point is in the territory being monitored by the network. For both models, we say such sampling points are *generated* by the node. If all the sampling points generated by every node in a subset are covered by the nodes in the same subset, the network territory is considered covered by the subset.

2.6.2 Algorithms in Comparison with MAXINE

For comparison purpose, we also implement two algorithms in solving Problem 2.1. The first is a Greedy Algorithm (GA) that selects an ungrouped node to the current subset (initially, the subset is an empty set $\varnothing$). The criterion is that adding this node causes the highest improvement of subset's the coverage values of the sampling points in total. This process is repeated until the subset can cover all the sampling points. The subsets are thus found iteratively. Based on the similar way we implement the distributed and localized version of MAXINE, we also

implement the Distributed and Localized GA (DLGA).

The second algorithm formulates the network partitioning problem as a disjoint set covers (dsc) problem, which can be transformed to a mixed integer programming (MIP), as shown in [13]. The subsets can be obtained with an MIP solver. For the Boolean coverage model, such an idea can directly be applied to the coverage-oriented network partition problem: By regarding each sampling point as a target to be covered, an MIP-based algorithm (MIPA) is thus implemented.

2.6.3 Simulation Settings

We randomly deploy sensor nodes in a $400m \times 400m$ area with a uniform distribution. The territory to be monitored by the network is a $360m \times 360m$ area centered at the $400m \times 400m$ area. R, β, δ, ϵ, and τ are 80.0, 2.0, 1.0, 100.0, and 0.6 respectively, which are some example values taken in quantifying the coverage for the collaborative probabilistic coverage model. R and τ are $60m$ and 1 for the Boolean coverage model. We conduct each of our simulations 10 times with different random seeds and the results are averaged. The simulations for the centralized algorithms are performed at a Sun Blade 2500 computer with two 1.6GHz UltraSPARC-IIIi CPUs and 2GB RAM. We

employ ILOG CPLEX Interactive Optimizer 9.1.0 to solve the MIP problem in MIPA.

The algorithm name, followed by the name of the coverage sampling method, indicates the scheme we employ to solve Problem 2.1. For example, MAXINE-HTS means that the MAXINE algorithm is employed and the coverage sampling method is HTS.

To quantitatively study the performance of the subsets found by each scheme, we randomly select 10000 event locations at the $360m \times 360m$ network territory in a uniform manner. We obtain the coverage of a subset at the event locations based on the coverage models. Each location where the coverage is below the requirement τ is considered as an event detection failure. We calculate the number of such event detection failures so as to study the event detection capability of a subset.

2.6.4 Comparisons of Coverage Sampling Methods

We compare the coverage sampling methods (UDRP, RL, HTS, and HMLS) in terms of the percentage of event-detection failures of the subsets found. The total node number is 200 and we vary the number of sampling points from 16 to 100. Figure 2.5 demonstrates the results by the centralized algorithms GA,

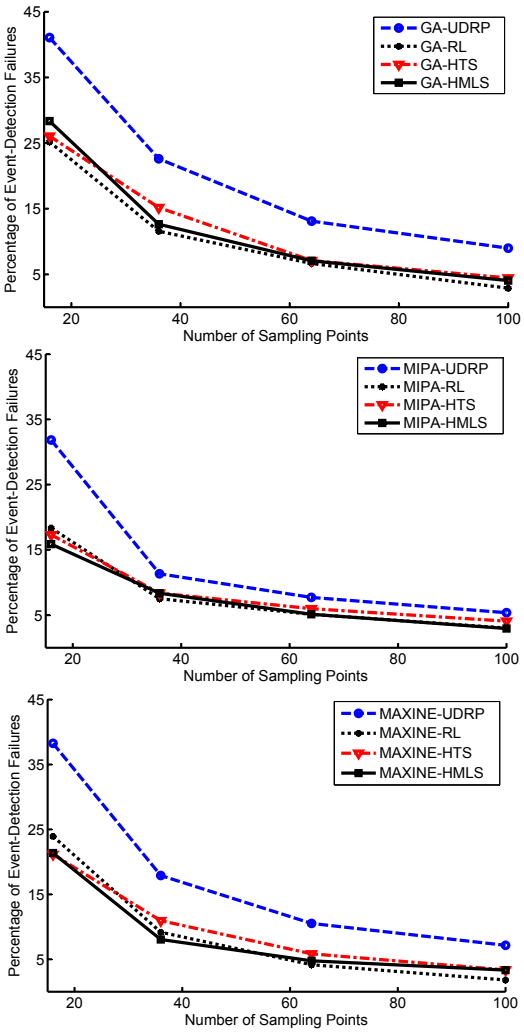


Figure 2.5: Comparisons of UDRP, RL, HTS, and HMLS

MIPA, and MAXINE. An instant result is that the number of the sampling points should be large enough (> 60) to maintain a low level ($< 10\%$) of event-detection failures. Furthermore, an algorithm with the RL, HTS, and HMLS methods performs much better than that with the UDRP method.

The good performance of the RL method is not surprising

because the RL method generates sampling points in a regular manner in which each sampling point can be regarded as the expectation of the event locations, which are uniformly distributed, in a small region. Discrete sampling points generated with quasi-random methods like HTS and HMLS have low discrepancy. As a result, the performance of an algorithm with a quasi-random method is also good. Because of the quasi-random nature of the HTS and HMLS methods, they can be good alternatives to the RL method, especially in the application cases that sensor nodes are deployed with some deterministic scheme (*e.g.*, grid-based deployment), where the sampling points generated by RL method may be with high risk in providing wrong coverage information as the pattern of sensor nodes and the pattern of sampling points may be correlated.

Note that we do not study the SNB method here because the number of sampling points generated by this method is not deterministic and consequently not comparable with the other methods.

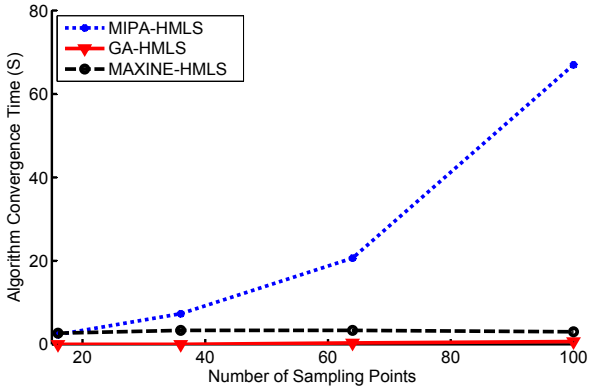


Figure 2.6: Impact of sampling-point number on convergence time

2.6.5 Impact of Sampling-point Numbers on Convergence Time

To study how the number of the sampling points influences the convergence time of the algorithms[4] that solve Problem 2.1, we deploy 250 sensor nodes in the network area, and vary the number of sampling points from 16 to 100.

The results are shown in Figure 2.6 where the HMLS method and the Boolean coverage model are adopted.

It shows that as the sampling-point number increases, the convergence time of MIPA increases quickly, while that of MAXINE and that of GA remain low in an almost constant manner. This is because the size of the MIP formulated in MIPA increases very quickly as the number of the sampling points increases. Note that we conduct these simulations with an advanced com-

[4]We consider the time in seconds required by the computer to complete an algorithm as the convergence time of the algorithm.

mercial MIP solver. For comparison purpose, we have done simulations with GLPK [37], a well-known open-source solver. The situation gets much worse when the number of the sampling points increases. On the other hand, in MAXINE and GA, the sampling points are involved only in testing whether a node should be selected/removed from a subset. This is not a major computational burden. As a result, the sampling-point number does not have a considerable impact on the convergence time.

2.6.6 Performance of the Centralized Implementation of MAXINE

We compare the centralized version of MAXINE with MIPA in terms of the number of subsets found. Since MIPA accepts only the Boolean sensing model, we adopt this model in this study. We vary the total number of sensor nodes n from 75 to 175 to see the impact of node density. The results are presented in Table 2.1, where x denotes the number of sampling points generated by the HMLS method. In Figure 2.7, we also show the impact of the total node numbers on the convergence time of these two algorithms, where the number of sampling points is 196.

From Table 2.1, we can see that when the node number is

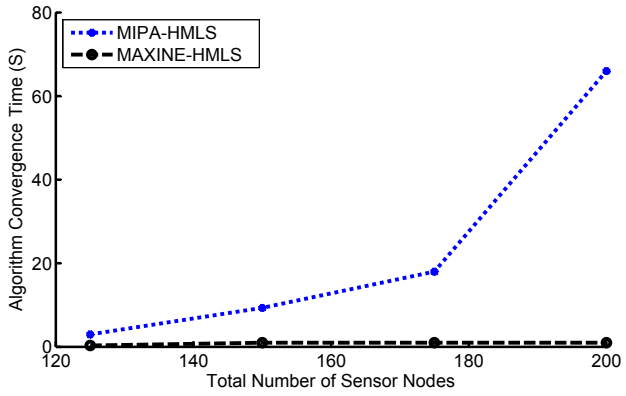


Figure 2.7: Impact of total node numbers on convergence time

small, the number of subsets found by these two algorithms are almost the same. When the node number is large, MIPA performs slightly better. However, in the large node number case, the convergence time of MIPA increases quickly as the total number of sensor nodes increases as shown in Figure 2.7. The reason is that the size of the MIP formulated in MIPA again increases very quickly as the node number increases.

We also investigate how the subsets perform in terms of the percentage of event-detection failures of the subsets. Figure 2.8 demonstrates the results in the case that the number of sampling points is 196. It shows that the performances of the subsets found by these two algorithms are comparable.

To compare GA and MAXINE, we vary the total number of sensor nodes from 100 to 300, and compare the number of subsets found by them. Figure 2.9 demonstrates the results, where

Table 2.1: Number of the subsets found by MIPA and MAXINE

	$x = 16$		$x = 36$	
n	MIPA	MAXINE	MIPA	MAXINE
75	2.00	2.00	1.67	1.33
100	4.00	4.00	4.00	3.33
125	5.33	5.33	4.67	4.33
150	7.67	7.33	6.67	5.33
175	9.00	8.33	8.67	6.67

	$x = 64$		$x = 100$	
n	MIPA	MAXINE	MIPA	MAXINE
75	1.33	1.33	1.33	1.33
100	2.67	2.67	3.00	2.67
125	4.00	3.33	3.67	3.00
150	6.00	5.00	5.67	4.67
175	7.00	6.00	7.33	5.00

	$x = 144$		$x = 196$	
n	MIPA	MAXINE	MIPA	MAXINE
75	0.67	0.67	1.00	1.00
100	3.00	2.67	3.00	2.33
125	3.67	2.67	4.00	3.00
150	6.00	4.67	6.00	4.67
175	6.00	4.33	6.33	4.67

the SNB method is employed and each sensor node generates 18 sampling points. It shows that MAXINE performs much better than GA in terms of the number of subsets found. Lastly, the percentage of event-detection failures of the subsets found in this study is less than 0.1%, which is very low, and hence is not included for comparison.

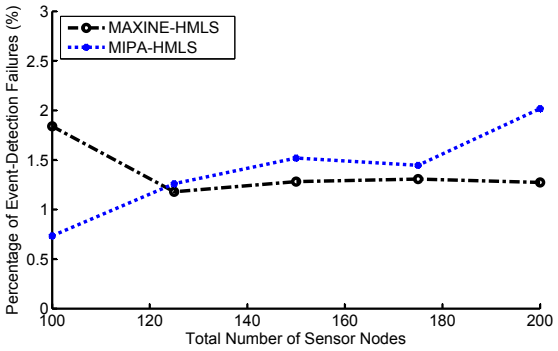


Figure 2.8: Event-detection failures of the subsets found by MIPA and MAXINE

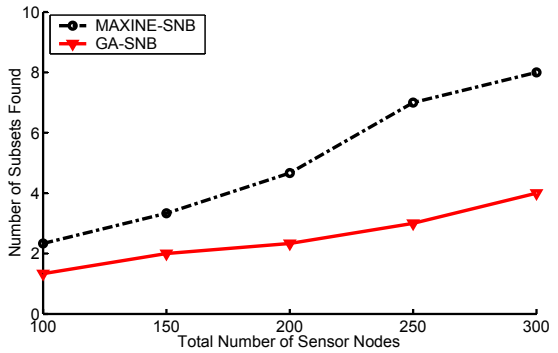


Figure 2.9: Number of the subsets found by GA and MAXINE

2.6.7 Performance of the Distributed and Localized Implementation of MAXINE

Since MIPA cannot easily be implemented as a distributed and localized algorithm, for fairness consideration, we compare the distributed and localized implementation of the MAXINE algorithm with DLGA in this study[5]. We adopt the collaborative probabilistic model and the SNB sampling method, where each node generates 12 sampling points.

[5]This part is based on my M.Phil. Thesis [126, 136].

Table 2.2: Results of five networks with n=1500

Net	MAXINE # of Subsets	DLGA # of Subsets	MAXINE Average ι	DLGA Average ι
1	34	31	0.145514	0.031702
2	33	30	0.145036	0.036649
3	33	31	0.156483	0.033578
4	32	31	0.152671	0.029030
5	33	32	0.146560	0.033109

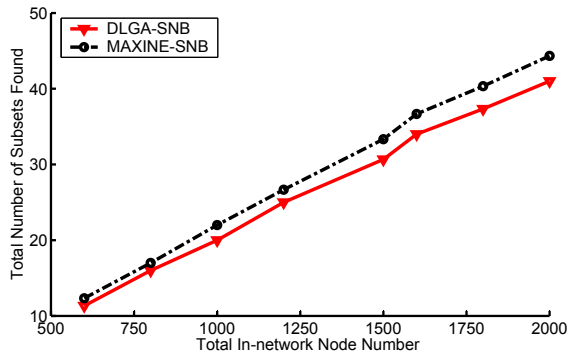


Figure 2.10: Number of subsets found by DLGA and MAXINE

We set the total in-network node number to different values and let the networks perform MAXINE and DLGA. Figure 2.10 shows the subset numbers found by both algorithms. We can see that MAXINE always outperforms DLGA in terms of the number of subsets found. Detailed results of five randomly-selected networks when $n = 1500$ are listed in Table 2.2. We find that the subsets formed by MAXINE always have larger ι values.

Figure 2.11 shows the average numbers of failure event detections when different node numbers are set. We can see that the

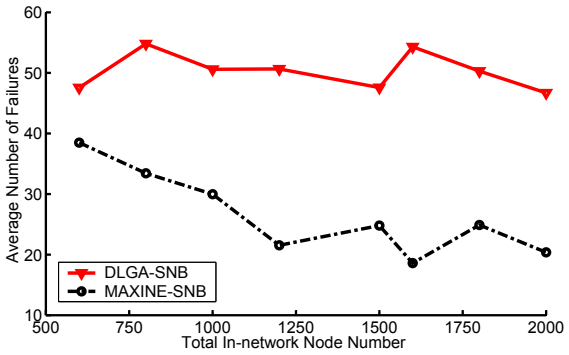


Figure 2.11: Event-detection failures of the subsets found by DLGA and MAXINE

subsets found by MAXINE outperform those found by DLGA.

All these simulation studies show that the ι-Thesis-based MAXINE algorithm, which maximizes the ι fan-out index of the subsets, exhibits very satisfying results in terms of the convergence time, the event detection capability of the subsets, and the number of subsets found.

2.7 Conclusion

This chapter showed the interesting features of the fan-out index ι for a set of points. The application of ι to the coverage-oriented network partition problem was elaborated by employing it in a MAXINE algorithm. MAXINE exhibited good performance and short convergence time in our extensive simulation studies with a wide range of network settings and two general sensor coverage models. This provides a strong evidence that maximizing ι is

a promising idea to reduce the node coverage redundancy. We will further discuss its application to another coverage-oriented problem in the next chapter.

□ **End of chapter.**

Chapter 3

Network Reconfiguration for Downtime-free System Migration

Summary

Many state-of-the-art WSNs have been equipped with reprogramming modules, *e.g.*, those for software/firmware updates. System migration tasks such as software reprogramming, however, will interrupt normal sensing and data processing operations of a sensor node. Although such tasks are occasionally invoked, the long time of such tasks may disable the network from detecting critical events, posing a severe threat to many sensitive applications. In this chapter, we present the first formal study on the problem of downtime-free migration. We demonstrate that the downtime can be effectively eliminated, by partitioning the sensors into subsets, and let them perform migration tasks successively with the rest still performing normal services. We then present a series of effective algorithms, and further extend our solution to a practical distributed and localized implementation. The performance of these algorithms has been evaluated through extensive simulations, and the results demonstrate that our algorithms achieve good balance between the sensing quality and system migration time.

3.1 Overview

It is usually expensive, if not impractical, for human-attended operations on a sensor node (especially for the WSNs applied in battle-field or habitat monitoring [70]). As a result, in most application scenarios, WSNs are expected to work in an unattended manner for a long period of time (usually several months) once the sensor nodes have been deployed.

Although sensing/processing environmental data is the major task of a WSN, during its month-long lifetime, it is inevitable for the WSN to perform certain system tasks in addition to this sensing/processing task, *e.g.*, system maintenance [12], diagnosis [17], and upgrading [110]. However, typical sensor platforms are simple and low-cost in nature. Multi-thread is usually not supported due to the capacity limitation of a sensor node [64]. As a result, these system tasks are exclusive to the data sensing/processing operations, *i.e.*, a sensor node has to cease data sensing/processing operations in performing these tasks. The most critical example of such system tasks is the WSN reprogramming task, which typically takes a network with one hundred sensor nodes a few hundred seconds to complete [110]. We

use the term *system migration tasks* to include these long-term exclusive system processes for reconfiguring, upgrading, or re-initializing existing network components or software/firmware, *e.g.*, reprogramming a sensing software-unit or re-initializing a communication protocol.

Although there have been significant research efforts on implementing efficient online migration tasks for WSNs [110], they largely ignored the interruption of the major sensing/repoting task caused by such a system migration task. This can be a critical threat in many application scenarios. For example, if a WSN for fire or intruder detection is being reprogrammed, it may fail to detect and alarm a fire/intrusion event which happens during the process. A reprogramming interval of hundreds of seconds is long enough to cause severe problems. Regular system migration tasks that occur periodically would further open this back-door for intruders to explore. Hence, it is very important to develop a seamless scheme for performing the exclusive system tasks, which avoids the downtime of normal network operations so as to maintain the uninterrupted event detection functionality of the network.

A natural way for downtime-free system migration is to divide the network into several subsets and let the subsets perform

the exclusive system migration task in turn, while the rest of the subsets still remain normal operations in sensing and processing environmental data[6]. Obviously, the more the number of subsets is, on one hand, the longer the time is to finish the system migration task for the whole networks; on the other hand, the less the performance degrades during the system migration. The number of the subsets thus can serve as a flexible parameter to fine-tune the system migration process. Given this number, the critical problem then becomes how the sensor nodes are partitioned into subsets so as to achieve the best tradeoff between system migration time and performance degradation.

Although various sensor grouping problems have been studied (*e.g.*, work in [13, 96], and that presented in Chapter 2 of this thesis), their objectives are generally to maximize the number of the subsets while maintaining the performance of each subset. This is quite different from the problem context here, and the conventional algorithms thus cannot be applied. In this chapter, we formulate the new *sensor network reconfiguration problem* for downtime-free system migration. We prove that the problem is NP-hard with a general probabilistic sensing model. We then

[6]We assume that the nodes before conducting the migration task and those after the conducting the task can still work collaboratively in event detection and notifying the sink when an event of interest is detected. This requires that the system before migration and that after migration are compatible.

present a series of heuristics and further extend one of them to a distributed and localized implementation. The performance of these algorithms have been evaluated through extensive simulations, and the results demonstrate that our algorithms achieve satisfactory balance between the sensing quality and the system migration time.

The rest of this chapter is organized as follows. Section 3.2 briefly introduces the related work. In Section 3.3, we provide a formal description of this problem. We then analyze this problem and prove its NP-hardness. Section 3.4 provides several algorithms in attacking this problem. The performance of these algorithms is studied in Section 3.5. We conclude this paper in Section 3.6.

3.2 Related Work

Implementing system migrations such as reprogramming is crucial to the success of WSN applications, with which bugs can be eliminated and functionalities of a network can be updated. See a survey paper and the references therein [110]. However, existing work has not notified the problem that the major sensing/repoting task would be interrupted by a system migration task. Our proposal is to divide sensor into subsets and let

each subset perform the migration task in turn. This scheme is then orthogonal to a reprogramming approach. It can be easily adopted in many existing reprogramming protocols, especially those that support the reprogramming of a selected group of nodes (*e.g.*, [73]).

Much work has been done on how to divide the sensor nodes in a network into disjoint subsets, where each subset can maintain the required sensing tasks. In [96], Slijepcevic *et al.* proposed to divide a sensing field into regions. Sensor nodes are grouped with the most-constrained least-constraining algorithm, in which the priority of selecting a sensor to a subset is determined by how much uncovered area this sensor covers and the redundancy caused by this sensor. In [14], the problem was modeled as disjoint dominating sets, which is known as NP-complete. A graph-coloring based approximation was then proposed. A similar problem of covering target points was studied in [13], which is again NP-complete and mixed integer programming (MIP) approximation has been proposed. In Chapter 2 of this thesis, we solved this problem by greedily maximizing the ι index of each subset. These approaches, however, are unapplicable to the problem studied in this chapter, as the objectives of these two problems are generally different.

3.3 Models and Problem Formulations

3.3.1 Preliminaries

We consider a WSN composed of n stationary sensor nodes $\{s_i\}_{i=1}^{n}$ randomly deployed in a uniform manner in a network area ϕ. Let the status of sensor node s_i be denoted by a binary variable c_i. c_i is 1 if s_i is conducting event detection work, and 0 if s_i is performing a system migration task such as being reprogrammed. For the convenience of our discussion, we say that a sensor s_i is *on* if $c_i = 1$, and *off* if $c_i = 0$. We call a collection of sensors a *division*, which can be represented by a sequence of n binary variables. For example, c_i $(i = 1, 2, ..., n)$ can represent a division D, which contains s_i if and only if $c_i = 1$, *i.e.*, all the on-sensors.

We consider a general probabilistic sensing model [66] as follows. The probability that an event e can be detected by a sensor s_i is related to the distance between e and s_i if the sensor is on; otherwise, it is zero since off-sensors can never detect an event. As the location of each s_i is fixed, the probability is determined by the event location (x, y) when the sensor is on. Let $p_i(x, y)$ denote this probability. Given the location of an event (x, y), the probability that the event can be detected by at least

one sensor in D is:

$$p_D(x, y) = 1 - \prod_{i=1}^{n}(1 - c_i p_i(x, y)). \tag{3.1}$$

Note that the network is employed to detect events and the events of interest can take place in any random location of the network area ϕ. As a result, the minimum value of $p_D(x, y)$ among all in-network locations (x, y) is a natural index to capture how badly the network division D might perform in event detection, which can then be considered as the capability of the network division D on event detection. Formally, we define that the *event detection capability* P_D of the network division D is the minimum value of $p_D(x, y)$ among all locations (x, y) in the entire network area ϕ, *i.e.*,

$$P_D = \min_{\forall(x,y)\in\phi} p_D(x, y). \tag{3.2}$$

This is a pessimistic measure, in which we pick the worst case as the representative case.

3.3.2 Problem Formulation

We consider that the network should be divided into N subsets, denoted by S_k $(k = 1, 2, ..., N)$. In order to enforce downtime-

free system migration, let each of the subsets perform the exclusive system migration task in turn, while the rest of the subsets still remain normal operations in sensing/processing environmental data.

Let d_{ik} denote whether s_i is in subset S_k. d_{ik} is 0 if s_i belongs to S_k, and 1 otherwise. In other words, when subset S_k is off (that a subset is off/on means all the sensors in the subset are off/on), sensor s_i is on if and only if $d_{ik} = 1$. So actually each sequence of d_{ik} $(i = 1, 2, ..., n)$ denotes the on-duty division D_k during the system migration of sensors in S_k, *i.e.*, let:

$$D_k = \{s_i\}_{i=1}^{n} - S_k. \tag{3.3}$$

In other words, D_k denotes the division of the sensor nodes which are conducting normal sensing and processing work when S_k is performing a migration task. d_{ik} then denotes whether s_i is in D_k.

N is naturally a parameter of such a downtime-free migration scheme. On one hand, a larger subset number means the longer time it requires for the whole network to finish the system migration task as the task is performed by each subset in turn. On the other hand, a larger subset number also means there are a smaller number of nodes in each subset. Since each time only

a subset is off in performing event detection task, fewer nodes in each subset implies that the system performance (in terms of event detection capability) degrades less during the system migration task.

The number of subsets N can thus serve as a flexible parameter for a system maintainer to fine-tune the system migration process. Considering the tradeoff between the system migration time and how much a system can tolerate the degrading of event detection capability during a system migration task, a system maintainer can determine how many subsets the network should be divided. Given an N by the system maintainer, the critical problem then becomes how the sensor nodes should be partitioned into subsets so as to achieve the best tradeoff between system migration time and performance degradation.

Suppose during a system migration, the network has to be reconfigured into N disjoint subsets S_k $(k = 1, ..., N)$ and let each D_k work successively. The *event detection capability of the entire network* in this migration interval is defined as the minimum among the event detection capability values of all the N divisions D_k $(\forall k)$. This actually continues the pessimistic considerations as how P_D is defined in Equation (3.2).

The sensor network reconfiguration problem is thus how to

divide the sensors so that the event detection capability of the network in this migration interval is maximized. Given the event detection capability measure P_D in Equation (3.2), the problem can be formulated as follows.

Problem 3.1: *The sensor network reconfiguration problem.*

Given: $\{s_i\}_{i=1}^n$ and $p_i(x, y)$

Partition the set $\{s_i\}_{i=1}^n$ into N disjoint subsets S_k $(k = 1, 2, ..., N)$ such that:

$$P = \min_{\forall k} P_{D_k} = \min_{\forall k}\{\min_{\forall (x,y)\in\phi}[1 - \prod_{i=1}^{n}(1 - d_{ik}p_i(x, y))]\}$$

is maximized. □

3.3.3 Difficulty of the Sensor Network Reconfiguration Problem

Given the situation that the number of the schemes to group n into N subsets is related to the permutation of n, while the event location is a continuous variable taken a value in the whole network area, the network location that results in $\min_{\forall k} P_{D_k}$ can be anywhere in the network. This makes it difficult to handle the sensor network reconfiguration problem.

Instead of dealing with all the network locations, let us confine our considerations such that events can only take place at a

finite set of discrete points in the network area. Suppose we have m such discrete points in the network area, denoted by $\{t_j\}_{j=1}^{m}$. We call them the *sampling points* of the network area ϕ. The probability that an event taking place at t_j can be detected by s_i is then denoted by $c_i p_{ij}$.

The minimum probability value among the sampling points, denoted by P'_D, is then:

$$P'_D = \min_{\forall j}[1 - \prod_{i=1}^{n}(1 - c_i p_{ij})]. \tag{3.4}$$

This P'_D thus serves as a simplified measure of the event detection capability P_D for division D. Then the sensor network reconfiguration problem turns to a *simplified* one given the practical measure P'_D in Equation (3.4), which can be formulated as follows.

Problem 3.2: *The simplified sensor network reconfiguration problem.*

Given: $\{s_i\}_{i=1}^{n}$, $\{t_j\}_{j=1}^{m}$, and p_{ij} $(\forall i, j)$

Partition the set $\{s_i\}_{i=1}^{n}$ into N disjoint subsets S_k $(k = 1, 2, ..., N)$ so that:

$$P' = \min_{\forall k} P'_{D_k} = \min_{\forall k}\{\min_{\forall j}[1 - \prod_{i=1}^{n}(1 - d_{ik} p_{ij})]\}$$

is maximized. □

Unfortunately, even this simplified problem is generally NP-Hard. To prove this, let us consider the decision version of this problem in which given the same problem settings, it asks whether the set $\{s_i\}_{i=1}^{n}$ can be partitioned into N disjoint subsets so that the event detection capability of the network in the migration interval is not smaller than a given value u. If the decision version of this problem is NP-Complete, then the sensor networks reconfiguration problem is NP-Hard [42]. In fact, we have the following lemma to prove the NP-Hardness of the simplified sensor network reconfiguration problem.

Lemma 1 *The decision version of the simplified sensor network reconfiguration problem is NP-Complete.*

Proof: First, this problem is in NP: Given a partition scheme, a nondeterministic algorithm only needs to calculate the event detection capability (EDC) of each division so as to get the EDC of the network during the time interval T. And then it can verify whether this value is smaller than u or not. So now we need to prove that this problem is harder than a known NP-Complete problem.

We transform the provably NP-Complete *set partition problem* [42] to the decision version of the simplified sensor network

reconfiguration problem. Given a set of non-negative numbers $\{q_i\}_{i=1}^{n}$, the set partition problem asks whether it is feasible to partition the set into two so that the sum of numbers in either partition is equal.

As $p_{ij} \in [0, 1)$, we can construct an n by m matrix $\mathcal{Q}$ of which each element is defined as $q_{ij} = -log_2(1 - p_{ij})$. We can know $q_{ij} > 0$. Based on the property of d_{ik}, we get:

$$1 - d_{ik}p_{ij} = 2^{-d_{ik}q_{ij}}. \tag{3.5}$$

Let us construct an instance of the sensor network reconfiguration problem in which $N = 2$, q_{ij} is equal to each other given the same i and equal to the q_i in the set partition problem, and $u = 1 - 2^{-(\sum_{i=1}^{n} q_i)/2}$. Now we can always have $d_{i1} = 1 - d_{i2}$ because a sensor should be in either division D_1 or division D_2, but not in both. Also we can write q_{ij} as q_i without the subscript j. Therefore, we get:

$$\begin{aligned} P' - u &= \min_{\forall k}\{\min_{\forall j}[1 - \prod_{i=1}^{n}(1 - d_{ik}p_{ij})]\} - u \\ &= \min_{\forall k}[\min_{\forall j}(1 - 2^{-\sum_{i=1}^{n} d_{ik}q_{ij}})] - u \\ &= \min_{\forall k}(1 - 2^{-\sum_{i=1}^{n} d_{ik}q_i}) - u \\ &= 2^{-\frac{\sum_{i=1}^{n} q_i}{2}} - 2^{-[\min_{\forall k}(\sum_{i=1}^{n} d_{ik}q_i)]}. \end{aligned} \tag{3.6}$$

If the answer to whether $P' \geq u$ is *yes*, we get:

$$\min_{\forall k}(\sum_{i=1}^{n} d_{ik}q_i) \geq \frac{\sum_{i=1}^{n} q_i}{2}$$

$$\Rightarrow \begin{cases} \sum_{i=1}^{n} d_{i1}q_i \geq \frac{\sum_{i=1}^{n} q_i}{2} \\ \sum_{i=1}^{n} d_{i2}q_i = \sum_{i=1}^{n}(1-d_{i1})q_i \geq \frac{\sum_{i=1}^{n} q_i}{2} \end{cases} \Rightarrow$$

$$\frac{\sum_{i=1}^{n} q_i}{2} \geq \sum_{i=1}^{n} d_{i1}q_i \geq \frac{\sum_{i=1}^{n} q_i}{2} \Rightarrow \sum_{i=1}^{n} d_{i1}q_i = \frac{\sum_{i=1}^{n} q_i}{2} \quad (3.7)$$

Therefore, the answer to the set partition problem is also *yes*.

On the other hand, if the answer to the set partition problem is *yes*, in the same way we can partition the sensors in the simplified sensor network reconfiguration problem so that:

$$\begin{cases} \sum_{i=1}^{n} d_{i1}q_i = \frac{\sum_{i=1}^{n} q_i}{2} \\ \sum_{i=1}^{n} d_{i2}q_i = \sum_{i=1}^{n}(1-d_{i1})q_i = \frac{\sum_{i=1}^{n} q_i}{2} \end{cases}$$

$$\Rightarrow \min_{\forall k}(\sum_{i=1}^{n} d_{ik}q_i) = \frac{\sum_{i=1}^{n} q_i}{2}. \quad (3.8)$$

According to Equation (3.6), $P' = u$. Therefore, the answer to the decision version of the simplified sensor network reconfiguration problem is also *yes*.

The above reduction requires only $O(n)$ steps to be completed

(for calculating p_i and u with q_i). Therefore, the decision version of the simplified sensor network reconfiguration problem is both NP-Hard and NP. Then it is NP-Complete. The lemma is proved.□

3.4 Heuristics for Downtime-Free System Migration

Given the difficulty of the sensor network reconfiguration problem, we resort to heuristics that can find the approximation solutions efficiently. We start from investigating this question: What should we make the resulting subsets look like, if we want to design a good approximation algorithm?

Let us again consider the simplified sensor network reconfiguration problem (Problem 3.2) first. For each solution to this problem, there exists one sample point t_x where the event detection capability of some division results in the minimum value, *i.e.*,

$$x = \underset{\forall j}{\operatorname{argmin}}\{\min_{\forall k}[1 - \prod_{i=1}^{n}(1 - d_{ik}p_{ij})]\}. \tag{3.9}$$

Suppose division D_y results in the minimum event detection

capability at t_x, *i.e.*,

$$y = \underset{\forall k}{\operatorname{argmin}}[1 - \prod_{i=1}^{n}(1 - d_{ik}p_{ix})]. \tag{3.10}$$

Let r_k denote the event detection probability for each set S_k at t_x. We get:

$$r_k = 1 - \prod_{i=1}^{n}(1 - d_{ik}p_i(t_x)), \tag{3.11}$$

where $p_i(t_x)$ denotes the event detection probability of sensor i at location t_x.

The event detection probability for each division D_k at t_x, denoted by $p'_{D_k}(t_x)$, is then:

$$p'_{D_k}(t_x) = 1 - \prod_{l=1, l\neq k}^{N}(1 - r_l) = 1 - \frac{\prod_{l=1}^{N}(1 - r_l)}{1 - r_k}. \tag{3.12}$$

$\prod_{l=1}^{N}(1 - r_l)$ is a constant based on Equation (3.11) because

$$\prod_{l=1}^{N}(1 - r_l) = \prod_{l=1}^{N}\prod_{i=1}^{n}(1 - d_{il}p_i(t_x)) = \prod_{i=1}^{n}(1 - p_i(t_x)), \tag{3.13}$$

which is irrelevant to how we group the sensor nodes.

Then based on Equation (3.12), if the event detection probability of D_y is the minimum among all D_k, $1 - r_y$ should be

the smallest among all $1 - r_k$ $(\forall k)$. In other words, r_y, *i.e.*, the event detection probability of subset S_y at t_x, must be the largest among all the subsets S_k.

The larger the event detection probability of S_y at t_x, the smaller the event detection probability of D_y at t_x. To maximize the event detection probability of D_y at t_x, the event detection probability of S_y at t_x should be minimized. Therefore, a good heuristic algorithm should let r_y be as close as possible to the event detection probability of other subsets at t_x.

3.4.1 Greedy Algorithm (GA)

The above consideration can be directly applied to an algorithm that solves the simplified sensor network reconfiguration problem (Problem 3.2): After initially grouping nodes into each S_k, we can greedily move the nodes in subset S_y to other subsets so as to reduce r_y.

Algorithm 1 demonstrates the mechanism of this greedy algorithm (GA). It first randomly selects $\frac{n}{N}$ nodes for each subset S_k (line 2). Let p_{min} denote the minimum event detection probability among the event detection probabilities of any division D_k $(\forall k)$ at any sampling point (line 5). GA locates the sampling point t_x at which the event detection probability of some division

Algorithm 1 Greedy Algorithm (GA)

1: **Input:**
   $\{s_i\}_{i=1}^{n}$: the set of sensor nodes
   $\{t_j\}_{j=1}^{m}$: the set of sampling points
   $\mathcal{P}$: the n by m matrix, where each element p_{ij} denotes the event detection probability of s_i at t_j.
2: Randomly selects $\frac{n}{N}$ nodes for each subset S_k
3: $D_k \leftarrow \{s_i\}_{i=1}^{n} - S_k$
4: **repeat**
5: $p_{min} \leftarrow$ the minimum event detection probability among the event detection probabilities of any divisions at any sampling points
6: $t_x \leftarrow$ the sampling point at which the event detection probability of a division is p_{min}.
7: $D_y \leftarrow$ the division that results in p_{min} at t_x
8: $S_z \leftarrow$ the subset whose event detection probability is the minimum at t_x.
9: Move a node from S_y to S_z. A node is selected if it results in the largest improvement of p_{min} comparing with selecting any other node. Ties are broken arbitrarily.
10: **until** p_{min} cannot be further improved

(denoted by D_y) is p_{min} (line 6). Based on the above discussions, the event detection probability of S_y is the maximum among all S_k at t_x. Now suppose the event detection probability of S_z is the minimum among all S_k at t_x. GA improves p_{min} by moving a node from S_y to S_z, which results in the largest improvement of p_{min} (lines 7-9). p_{min} is thus improved iteratively until it cannot be further improved (lines 4-10).

Although Algorithm 1 is to solve Problem 3.2, with a set of well-designed sampling points $\{t_j\}_{j=1}^{m}$, its output can be deemed as a solution of Problem 3.1. The prerequisite is that the sam-

pling points can well represent the event detection probability of the whole network. Quasi-random sequences, such as Hammersley sequence, which possess asymptotically optimal discrepancy (a measure of uniformity for the distribution of the points) have been widely employed in Quasi Monte Carlo methods [46]. In this regard, they are reasonably good sampling-point generators. We adopt Hammersley sequence [46] to generate the sampling points for Algorithm 1. We linearly map the 2-dimensional Hammersley sequence into the network area to generate the locations of the sampling points $\{t_j\}_{j=1}^{m}$ as the input of the algorithm.

3.4.2 Simple Partitioning and Picking Algorithm (SPP) and Minimum Spanning Tree-Based Grouping Algorithm (MSTBG)

In the greedy algorithm, t_x is found in the set of sampling points. In fact, given the original settings of Problem 3.1, *i.e.*, the event location is a continuous variable which can be anywhere in the network, actually a possible t_x can also be anywhere in the network. To minimize r_y, it would therefore be better if all the subsets have a closer event detection probability at any point in the network as we do not know where t_x should be.

In order to make the event detection probability of all the

subsets close to each other at any points, the resulting subsets should look similar in a dispersive manner. It is therefore necessary that nodes in the same subset should be dispersedly distributed. Nodes that are close to each other should not be grouped into the same subset so as to avoid high event detection probability of the subset at locations around these nodes. In other words, if we examine an arbitrary area in the network, there should not be outstanding dominant-population of any one of the subsets.

Based on this consideration, we design the other two algorithms, namely, the Simple Partitioning and Picking (SPP) algorithm and the Minimum Spanning Tree-Based Grouping (MSTBG) algorithm.

SPP tries to maximize the ι index, *i.e.*, the minimum distance over the average distance between each node pair, of the resulting subsets. Because the ι index can serve as a good microscope in indicating the existing of a high redundancy area as we have discussed in Chapter 2, by maximizing this fan-out index, SPP aims at avoiding high redundancy of some subsets comparing to the others at anywhere in the network.

Algorithm 2 shows the details of SPP. It performs two procedures in turn: the partitioning procedure (lines 2-13) and the

Algorithm 2 Simple Partitioning and Picking (SPP)

1: **Input:** $\{s_i\}_{i=1}^{n}$: the set of sensor nodes
2: The whole network area is deemed as a region
3: **repeat**
4: **for all** regions **do**
5: draw a line parallel to the x-axis to partition the region into two so that the difference between the node numbers in both partitions is at most one.
6: **end for**
7: **if** there are more than $2N$ nodes in each region **then**
8: **for all** regions **do**
9: draw a line parallel to the y-axis to partition the region into two so that the difference between the node numbers in both partitions is at most one.
10: **end for**
11: **end if**
12: **until** there are less than $2N$ nodes in each region
13: randomly select nodes in each region to $S_k(k = 1, ..., N)$
14: **repeat**
15: randomly assign two neighboring regions as A and B
16: **for** each randomly-picked subset A_k in region A **do**
17: Couple A_k with a subset B_x in region B, such that the couple results in the largest ι comparing to the other couples formed by A_k and any other subsets in B. Ties are broken arbitrarily.
18: **end for**
19: each couple is deemed as a subset, and thus A and B are merged into a larger region.
20: **until** there is only one region

merging procedure (lines 14-20). In the partitioning procedure, first consider all nodes are in one region. Supposing there is a Cartesian coordinate system in the network area, SPP iteratively cuts each region into two until there are less than $2N$ nodes in every region (lines 4-12). Then nodes in each region are randomly selected into N different subsets (line 13). In the

merging procedure, neighboring regions are merged into one till there is only one region. During the merge procedure, one subset in a region is coupled to anther subset in another region if the couple can result in the largest ι comparing to the other possible couples. Ties are broken by picking randomly. This coupling process continues until all N couples are generated, since each region has N subsets (lines 16-18). Then each couple is deemed as a subset, and thus two neighboring regions are merged into one larger region (line 19).

Based on the same notion that nodes close to each other should not be grouped into the same subset, MSTBG constructs a minimum spanning tree (MST) of the network incrementally, and groups each newly-joining node to its farthest subset in the tree[7]. Thus MSTBG tries to group nodes that are close to each other to different subsets so as to avoid the close-gathering of the nodes in the same subset. The algorithm is illustrated in Algorithm 3.

MSTBG first builds a tree that is composed only by the two closest nodes among all the in-network nodes. These two nodes are grouped into two different subsets (lines 2-3). Select a node which is the closest to the tree among all the nodes that are not

[7]The distance between a node and a subset is defined as the distance between the node and its closest node in the subset.

Algorithm 3 Minimum Spanning Tree-Based Grouping Algorithm (MSTBG)

1: **Input:**
   $\{s_i\}_{i=1}^{n}$: the set of sensor nodes
2: $T \leftarrow$ the tree composed by two closest nodes.
3: group the two nodes into two arbitrary subsets.
4: **repeat**
5: $s \leftarrow$ the node closest to the tree among all the nodes that are not in the tree, where the distance between a node to the tree is defined as the minimum distance between the node to those nodes in the tree.
6: calculate the distance between s and each subset in the tree
7: $S \leftarrow$ the farthest subset (ties are arbitrarily broken)
8: group s to S, and add s to T
9: **until** all nodes are in T

in the tree (line 5). Group this node to the subset which is the farthest to this node. Then add this node to the tree (lines 6-8). This procedure is thus iteratively conducted until all nodes are in the tree. Such a process in building a tree is exactly how Prim's algorithm works in building a minimum spanning tree (MST) [84]. This is why we call this algorithm an MST-based grouping algorithm.

3.4.3 SNRP: A Distributed and Localized Sensor Network Reconfiguration Protocol

The algorithms discussed above (*i.e.*, GA, SPP, and MSTBG) are all centralized approaches. A global picture of the network is required to run these algorithms. However, WSNs are usually large-scale networks which contain hundreds of nodes. Global

information is not easy, if not impossible, to be obtained. According to the features of WSNs, a distributed and localized solution for the sensor reconfiguration problem is surely of practical interests.

Although some well-investigated mechanism can help implement the above algorithms in a distributed way, for example, a distributed MST algorithm (*e.g.*, [39]) can be applied to decentralize MSTBG, global information is still inevitably required in constructing an MST [65, 133]. We therefore design a new distributed and localized algorithm called the Sensor Network Reconfiguration Protocol (SNRP). It is based on a mechanism similar to that in MSTBG. But instead of constructing an MST of the entire network, in SNRP, each node builds its local MST of its neighborhood graph (*i.e.*, the graph consisting of the node and its one-hop neighbors). The nodes then groups neighboring nodes to the subsets based on the local MST.

SNRP is an event (packet) driven algorithm. There are four types of packets involved in this algorithm, *i.e.*, ASK, CANCEL, ANSWER, and RESULT packets. Figure 3.1 demonstrates SNRP with a finite state machine. Details on the protocol are as follows.

Initially, a node does not belong to any subset (S0). The

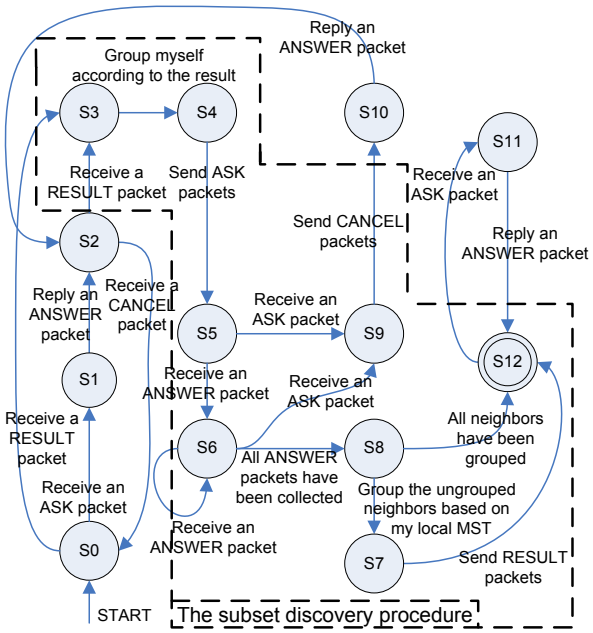


Figure 3.1: Finite state machine of SNRP

base station (*i.e.*, the network control center) will firstly send a RESULT packet to a randomly selected node, telling it that it belongs to subset 1 and let it begin to perform the subset discovery procedure.

When a node (suppose it is node s) receives a RESULT packet (S0→S3), it starts its subset discovery procedure by firstly sending ASK packets to enquire its neighbors which subset they belong to (S4→S5). It waits until every neighbor has replied with an ANSWER packet (S6→S8). Then node s constructs a local MST of its neighborhood graph. Based on the same mechanism as that in MSTBG, it groups each of the nodes which do not belong to any subset into a subset (S8→S7). Then node s

notifies these nodes the grouping results by sending them RESULT packets (S7→S12). Thus the subset discovery procedure is handed over to the neighboring nodes of node s and node s comes to the final state ($S12$).

If a node receives an ASK packet from a neighbor (suppose the neighbor is node s'), the node will behave differently according to whether or not it is currently conducting the subset discovery procedure. If this is true, *i.e.*, when the node is waiting for collecting all ANSWER packets in the subset discovery procedure (S5 or S6), in order to avoid deadlocks it will send CANCEL packets to all the neighbors to which it has sent ASK packets (S9→S10) before it sends an ANSWER packet to node s' in reporting which subset it belongs to (S10→S2). Otherwise, ($S0$ or $S12$), it will directly send an ANSWER packet to node s' (S1→S2 or S11→S12).

Then after sending the ANSWER packet to node s', the node will return to the final state if the node has successfully performed the subset discovery procedure before (S12). Otherwise, it waits for a RESULT packet or a CANCEL packet from node s' (S2). Note that the node will also queue the ASK packets from other neighbors without reply during this waiting time. This can avoid the node to be grouped into different subsets by

different neighbors. Now if a CANCEL packet is received, the waiting is canceled (S2→S0) and the node returns to the initial state (S0).

3.5 Performance Study

To study the effectiveness of our algorithms in solving the sensor network reconfiguration problem, we customize a sensor network simulator. Detailed settings of the simulation networks are shown in Table 3.1. β, δ and ϵ in the table are parameters of the probabilistic sensing model in which if an event is L meters away from a sensor, the sensor can detect the event with probability p that satisfies:

$$p = \begin{cases} \frac{\delta}{(L/\epsilon+1)^{\beta}} & \text{if } L \leq R_s, \\ 0 & \text{otherwise.} \end{cases} \tag{3.14}$$

This model implies that the event detection probability is determined by the event-signal strength received by a sensor, while the signal fades exponentially with a factor β in its way from the event location to the sensor location. This is a realistic consideration.

We employ SPP, GA, MSTBG, and SNRP to reconfigure the

Table 3.1: Simulation settings in Chapter 3

Area of sensor field	200m × 200m
Rode deployment scheme	Randomly deployed in a uniform manner
Sensing range R_s	40m
Communication range R_c	40m
β, δ and ϵ	2.0, 1.0 and 40.0
Number of sampling points	100
Sampling method	2-dimensional Hammersley sequence

in-network sensor nodes into N subsets. We study the event detection capability (EDC) of the network during a system migration task where each subset has to cease to work for a given period of time successively. For each network setting, simulations are performed for 100 times with different random seeds and the results are averaged.

For comparison purpose, we also draw another three curves. The first curve (named "Original" in the figures) shows the EDC of the entire network when no subset is off. The second curve (named "Upper Bound" in the figures) shows the EDC upper-bound of the network when one subset cease to work, which is computed by:

$$1 - (1 - P_{all})^{\frac{N-1}{N}} \tag{3.15}$$

where P_{all} is the EDC of the entire network when no subset is off. This is, however, a *non-achievable* upper-bound, as it considers the non-achievable but optimum case where each subset

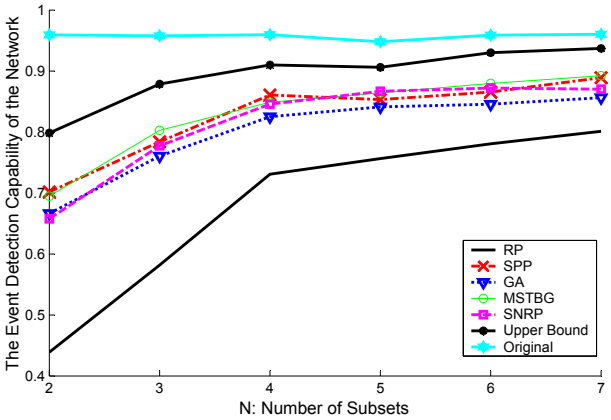


Figure 3.2: EDC as a function of N (Node Number = 100)

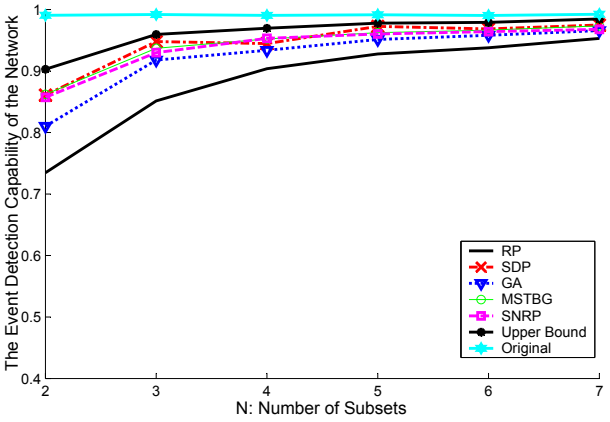


Figure 3.3: EDC as a function of N (Node Number = 150)

has equal event detection probability at any point of the network. Lastly, the third curve (named "RP" in the figures) is the EDC of the network when reconfigured by a Random Pick algorithm (RP), in which we randomly select n/N nodes for each subset without any performance considerations. This serves as a baseline in our simulation study.

We first study how the value of N influences our algorithms.

Figures 3.2-3.4 show the EDC of the networks composed of different numbers of nodes. We can see that the naive RP algorithm performs by far the worst, which is what we have expected. SPP, MSTBG, and SNRP always perform better than GA, which verifies that it is necessary to disperse the nodes in the same subset.

Also it can be found out that when N is large enough ($N > 4$ in our simulations), improving N cannot effectively improve the EDC of the network. This is not strange. The difference between the average node number in a subset when we divide the network in N subsets and that when we divide the network into $N - 1$ subsets gets smaller and smaller as N increases. When one subset is off, the difference of the EDC degradations in these two cases is hence become smaller. However, as a larger N incurs longer time for the entire network to complete a migration task, the price of improving the EDC of the network during system migration becomes higher and higher as N increases. This should be an important consideration for a system maintainer to select a proper value of N.

When the node density becomes higher, the differences among the performances of these algorithms become smaller. This is not surprising, either. The more the in-network nodes, the

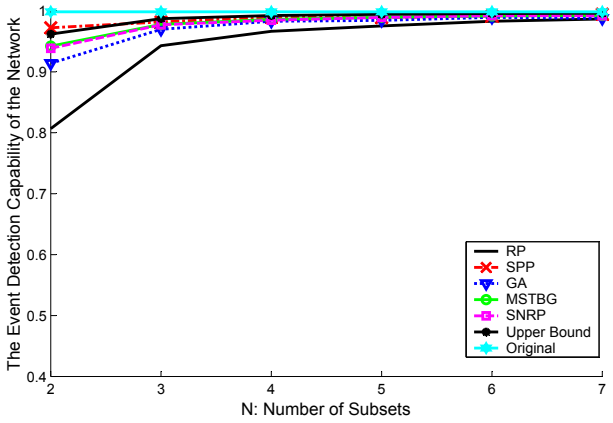


Figure 3.4: EDC as a function of N (Node Number = 200)

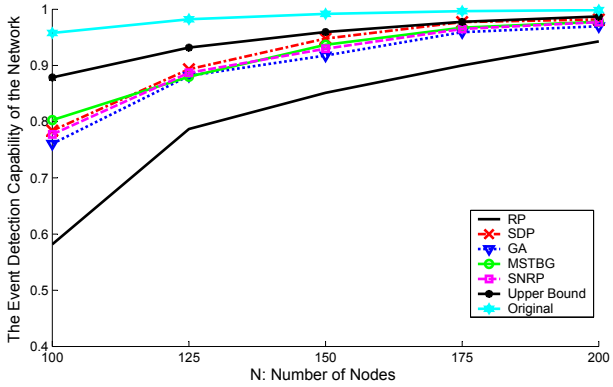
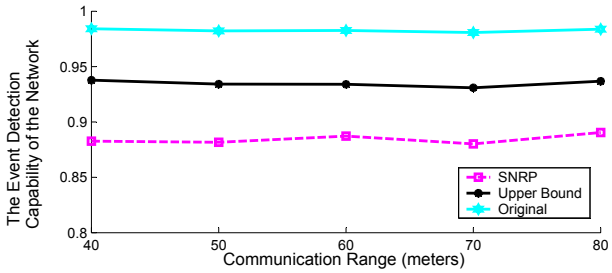


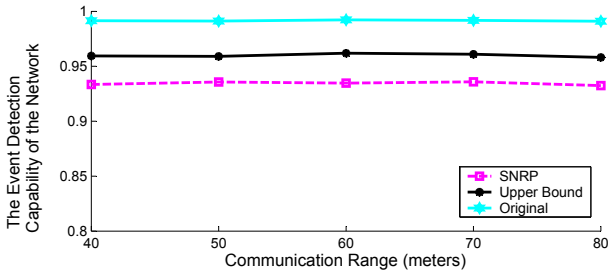
Figure 3.5: EDC as a function of node number

higher the redundancy of the network with respect to event detection. As a result, when the node density is high, if one subset of the nodes is off, it does not have a great impact on the performance of the network. Figure 3.5 further demonstrates this idea. We let $N = 3$ and change the number of nodes from 100 to 200. We can see that the EDC of the network gradually approaches the original curve.

To see how the neighborhood graph size influences the results



(a) Node Number = 125



(b) Node Number = 150

Figure 3.6: EDC as a function of communication range

of SNRP, we change the communication range R_c from $40m$ to $80m$ (*i.e.*, from one time to two times of the sensing range). Figure 3.6 demonstrates the experimental results of SNRP where $N = 3$ and the node numbers are 125 and 150, respectively. We can find out that SNRP performs almost the same when R_c is larger than the sensing range. As usually the communication range of a node is larger than its sensing range, these results verify that grouping based on the local MST is good enough comparing to grouping based on the global MST. It shows that SNRP, as a distributed and localized algorithm, works very well since it does not degrade the resulting performance much.

3.6 Conclusions

Seamless system migration without downtime is necessary for WSNs that perform critical event detection tasks. Unfortunately, to our knowledge, this important problem has not been addressed in the literature. In this chapter, we presented the first formal study on this problem. We demonstrated that the downtime can be eliminated by partitioning the sensors into a collection of subsets, and let each subset conduct the system migration tasks successively with the rest still performing normal event detection services. We proved the optimum partitioning of sensors in this context is NP-hard and then proposed a series of heuristics. We further extended our solution to a distributed implementation called the Sensor Network Reconfiguration Protocol (SNRP). Simulation results showed that these algorithms work well in various performance evaluations.

□ **End of chapter.**

Chapter 4

Surviving Holes and Barriers in Geographic Data Forwarding

Summary

Geographic forwarding is a favorable scheme for data reporting in Wireless Sensor Networks (WSNs) due to its simplicity and low-overhead. However, WSNs are usually subject to complicated environmental factors. Network holes (*i.e.*, the areas where no nodes inside) and barriers (*i.e.*, those blocking the communication between two close nodes) are inevitable in practical deploying environments. These issues pose an obstacle to adopting geographic forwarding in WSNs, while current approaches lack an efficient method to tolerate such negative factors. In this chapter we specifically tailor a waypoint-based Geographic Data Reporting Protocol (GDRP) for WSNs. Inherited from geographic forwarding, GDRP is light-weighted and hence well-suits WSNs. But unlike current approaches that often find suboptimal paths, GDRP adopts an intelligent strategy to select a best set of waypoints via which packets can efficiently circumvent holes and barriers, and it can thus find better paths. Extensive simulations are conducted to verify the advantages of GDRP in tolerating network holes and obstacles in WSNs.

4.1 Introduction

A crucial task for a WSN is to convey the sensor data to a sink so that the sink can obtain the information of interest. *Geographic forwarding* [9, 55, 56] is favorable for this sensor-to-sink data reporting task. Its basic idea, namely, *greedy forwarding*, is that a packet is forwarded to a neighbor which is geographically closer to the intended destination till the packet eventually reaches the destination. In geographic forwarding, each node just needs to maintain the information of its neighborhood to select the next-hop neighbor to forward packets. This provides it nice scalability and low overhead merits, which are specifically beneficial to WSNs due to their large-scale feature and the energy constraints suffered by sensor nodes [21].

However, network holes (*i.e.*, the areas where no nodes inside) and barriers (*i.e.*, those blocking the communication between two close nodes) are inevitable in practice [1]. Various real-world geographical environments, *e.g.*, the existence of puddles or buildings where sensors cannot be deployed causes holes. Hills, walls, or even trees may cut wireless link of two nodes even if they are in the theoretical communication range of each

other, and thus form barriers. All these can make greedy forwarding infeasible as a node may not find any neighbor nearer to a packet's intended destination than itself.

To tolerate the failure of greedy forwarding, traditional geographic forwarding schemes enter a *detour mode* in which a packet is sent to a neighbor farther to the destination but potential in bypassing a hole [9, 56]. The detour mode tends to forward data packets along the boundaries of holes. It usually turns out that the path from the source to the destination is much longer than the optimum [8]. Longer path incurs more energy consumption for packet transportation. It is therefore critical to enhance the survivability of geographic forwarding in practical deployment environments of WSNs, where there are many network holes and barriers.

In this chapter, we specifically tailor a waypoint-based Geographic Data Reporting Protocol (GDRP) for WSNs to address this challenging problem. The purpose of waypoint-based geographic forwarding is to minimize the unnecessary detours by forwarding packets along a sequence of waypoints so that the path can bypass holes and barriers. Lengthy routes due to the detour mode can thus be avoided. GDRP adopts a trial-and-error approach: The information of holes and barriers is accu-

mulated during the runtime of GDRP. Based on such information, better and better waypoint sequences can then be designed. Unlike the current state-of-the-art approaches [8, 51, 124] which may find waypoints that result in suboptimal path, GDRP can find an optimal path from the source to the sink. We formulate how to select waypoints as a tractable problem and provide its solution with details in handling realistic network situations. In contrast to the existing work [8, 51, 124] that provides heuristics in finding waypoints, we prove the performance guarantee of GDRP.

The rest of this chapter is organized as follows. Section 4.2 presents the related work. In Section 4.3, we provide the motivations of this research. We then sketch GDRP and identify its three key components. Section 4.4 elaborates the design of GDRP and illustrates how it can survive network holes and barriers energy-efficiently. We study the performance of GDRP with extensive simulations in Section 4.5. Finally, Section 4.6 concludes this chapter.

4.2 Related Work

4.2.1 Geographic Forwarding

Geographic routing is first proposed by Karnakis *et al.* in [55]. Greedy Perimeter Stateless Routing (GPSR) [56] and several other algorithms were subsequently proposed (*e.g.*, [9]), which adopt greedy forwarding if it is feasible, and otherwise enter the detour mode and perform face routing in a planar graph of the network. These algorithms can guarantee successful packet delivery. Frey and Stojmenovic further showed that recovery from a greedy forwarding failure is always possible without changing a face in the detour mode [38].

One challenge to geographic forwarding is that the locations of the nodes may not be known [36], when a node is node equipped with a GPS receiver. One possible approach is via a localization protocol (*e.g.*, [10, 114]).

Face routing incurs a longer path [8]. Many schemes are proposed to improve face routing. Fang *et al.* studied how to locate network holes and proposed to route packet along the boundary of a hole [34]. Leong *et al.* presented a geographic routing mechanism without planarizing the network [63]. These approaches generally need a protocol to obtain the information of the net-

work holes and aim to minimize the protocol overhead, resulting a more complicated implementation. They largely focus on holes and lack a scheme to handle network barriers. Furthermore, related work also includes those focusing on finding a geometric embedding of the network where greedy forwarding is always feasible [36, 62]); and those assigning the nodes virtual coordinates, via which data forwarding is conducted [102, 107].

Waypoint-based geographic forwarding was proposed in [8] and Huang [51] studied how to select a set of waypoints adaptively. But the waypoint selection scheme may be trapped to suboptimal results. Moreover, Zhao *et al.* [124] proposed to conduct random shift to the locations of the waypoints to avoid some nodes are always selected.

Theoretical performances of greedy forwarding has also been widely studied. For example, Wan *et al.* provided the asymptotic bounds of transmission range to ensure greedy forwarding [109]. Zorzi *et al.* provided the bounds on hop-count and latency performance of greedy forwarding [137, 138].

4.2.2 Data Transport in WSNs

Data transport in WSNs has long been studied in the literature. In reliability domain, early data transport protocols for

WSNs include PSFQ [108] and RMST [101]) that focus on end-to-end reliable data transport. Event-to-sink data transport was studied in [52, 104]. In [89], it was suggested that absolute end-to-end reliable data transport is usually not needed when transmitting sensor reporting packets. Packet loss within a certain limit can usually be well tolerated in most application scenarios. Based on this notion, an application-specific reliable data transport protocol was proposed in [132]. There are also many existing real-time data transport protocols in the literature for WSNs. Lu *et al.* [69] described a packet scheduling policy called Velocity Monotonic Scheduling. It accounts for both time and distance constraints. SPEED [47] by He *et al.* combines feedback control and non-deterministic QoS-aware geographic forwarding. Felemban *et al.* [35] proposed Multi-path and Multi-Speed Routing Protocol (MMSPEED) for probabilistic QoS guarantee in WSNs. Multiple QoS levels are provided in the timeliness domain by using different delivery speeds, while various requirements are supported by probabilistic multipath forwarding in the reliability domain. A multi-queue-based protocol was suggested in [77, 78], where soft QoS in timeliness domain and reliability domain is investigated.

The notion of transmitter power control (topology control)

has been extensively studied in MANETs. Most work (*e.g.*, [16, 86, 88, 90, 91, 112]) focused on network connectivity analysis, network lifetime (a network is alive if it is 'somehow' connected) analysis, where transmitter power setting schemes, generally for energy-efficient communications between an arbitrary node pair and for energy-efficient broadcasting and multicasting, were proposed. Work in [127] investigated transmitter power control for sensor-to-sink traffic and suggested an efficient waiting-based method for setting up such paths. Work in [134] suggested a location-directed protocol for transport real-time data for WSNs based on transmitter power control.

4.3 Waypoint-Based Geographic Forwarding

We consider a WSN consisting of a sink d and N stationary sensor nodes randomly deployed in a 2-dimensional network area ϕ (*i.e.*, $\phi \subset \mathbb{R}^2$). Since location-awareness is a basic requirement for geographic forwarding algorithms [9, 56], we consider that each node is aware of its own location, which can be obtained by GPS or a localization approach (*e.g.*, [10]). Let r denote the communication range of a node. Each node u can then know the locations of its neighbors, *i.e.*, those nodes that have a wireless link with u. Note that due to the existence of barriers

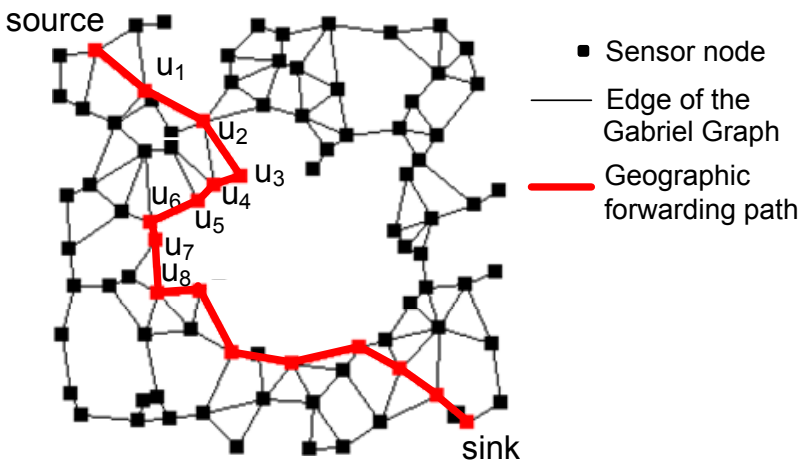


Figure 4.1: A geographic forwarding example

in practical working environments, two nodes may not have a wireless link even when their distance is less than r. In this case, they are not neighbors.

4.3.1 Geographic Forwarding

Geographic forwarding schemes [9, 56] firstly planarize a network into a planar graph G_P (*e.g.* Gabriel Graph or Relative Neighborhood Graph) with a distributed and localized algorithm as shown in Figure 4.1. They usually contain two working modes: *the greedy mode* and *the detour mode*. In the greedy mode, a node u selects a node v as its next hop among u's neighbors that are closer to the sink than itself when v is the closest one to the sink (*e.g.*, in Figure 4.1, the source selects u_1 and u_1 selects u_2). When holes and barriers exist, the greedy mode

may not be feasible, *i.e.*, no neighbor is closer to the sink (*e.g.*, for u_3). To handle this case, geographic forwarding schemes send packets in the detour mode: Packets are sent clockwise (or counter-clockwise) along the face of G_P which are closer to the sink (*e.g.*, u_3 selects u_4 as its next-hop and u_4 selects u_5 as its next-hop), until greedy forwarding is feasible again at a node (u_8) which is also closer to the sink than the node where greedy forwarding first got stuck (u_3) [56].

4.3.2 Design Considerations of Waypoint-based Geographic Forwarding

The design objective of a data reporting scheme is to minimize the energy consumptions of the packet transportation from the source to the destination. Since the hop number of a path shows how many nodes are involved in the packet transportation, it is a natural indicator to such energy consumptions. We define:

Definition 1 *The topological length of a path is the total number of hops between the source and the destination of the path.*□

A *waypoint sequence* is a sequence of sensor nodes that serve as intermediate relays for the packet transportation from the source to the destination. Let W=[w(1), w(2),..., $w(M)$] denote a waypoint sequence with size M where each element denotes

a waypoint node. Note that we deem the source as the first waypoint and the destination as the last waypoint. We define:

Definition 2 *The Euclidean length of a waypoint sequence* $[w(1), ..., w(M)]$ *is* $\sum_{i=2}^{M} \mathbb{D}(w(i), w(i-1))$*, where* $\mathbb{D}(\cdot, \cdot)$ *denotes the Euclidean distance between two nodes.*□

In *waypoint-based geographic forwarding*, packets from the first waypoint, *i.e.*, the source, will be sent to the second waypoint. A waypoint, when receiving a packet, will send the packet to its next waypoint until the last waypoint (*i.e.*, the destination) is reached. Packets are transported between two adjacent waypoints with a geographic forwarding scheme.

The aim of a waypoint-based geographic forwarding scheme is to find a waypoint sequence $[w(1), ..., w(M)]$ given a source $w(1)$ and destination $w(M)$ so as to minimize the topological length of the path from $w(1)$ to $w(M)$. To achieve this, two issues are generally considered. First, greedy forwarding should always be feasible between two adjacent waypoints based on the notion that the greedy mode is better than the detour mode in terms of the topological length. Second, the Euclidean length of the waypoint sequence should be minimized since longer Euclidean length usually incurs larger topological length. However, these two considerations in the existing approaches are not yet

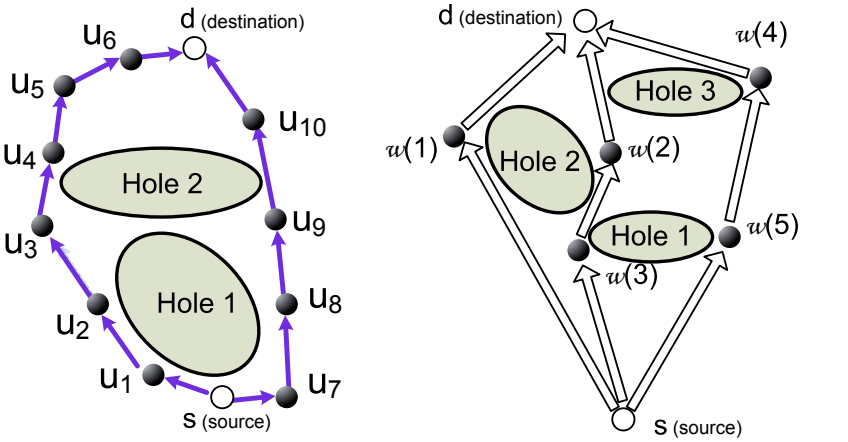


Figure 4.2: Examples of suboptimal results

adequate.

First, even if greedy forwarding is always feasible, geographic routing cannot always achieve optimal routes [51]. Figure 4.2(a) shows an example where a greedy forwarding path (*i.e.*, the left-hand side path) is just a suboptimal path. In fact, the topological length of greedy forwarding between s and d is tightly-bounded by $O(\mathbb{D}^2(s,d))$ [41]. Such a *quadratic* relation makes the theoretical performance of the greedy mode only comparable to the detour mode. We are motivated to enhance waypoint-based geographic forwarding so that the topological length between two adjacent waypoints is *linearly* related to their distance.

Second, current approaches lack a strategy that can calculate a waypoint sequence with minimum Euclidean length. Huang [51] has recently suggested that packets should be forwarded along the convex hull of the geometry set Z which consists of the source, the destination, and the holes and barriers between them. Supposing there is a 2-dimensional Cartesian coordinate system with its x-axis passing the source and the destination, this approach finds a waypoint sequence where the waypoints form the $y > 0$ or the $y < 0$ half of the convex hull of Z. However, this approach may not find a good waypoint sequence, since a shorter sequence can penetrate the set Z rather than going around it. Figure 4.2(b) shows an example where the waypoint sequence $[s, w(3), w(2), d]$ is shorter in terms of Euclidean length than the waypoint sequences $[s, w(1), d]$ and $[s, w(5), w(4), d]$ that can be found by this approach.

How can we find a shorter path than those that go around Z while the path can still bypass the holes and the barriers in Z? A brute-force strategy is that for each hole or barrier in front, packets are forwarded to both left-hand side and right-hand side of the hole or barrier. Euclidean lengths of all the possible resulting waypoint sequences are compared and the minimum one is selected. This strategy can find a best waypoint sequence

since it tries all options. However, the number of options is 2^m where m is the number of holes and barriers between the source and the destination. When there are many holes and barriers in between, the number of options is huge, which is surely unacceptable. Hence, a better approach that can converge in polynomial time is desired.

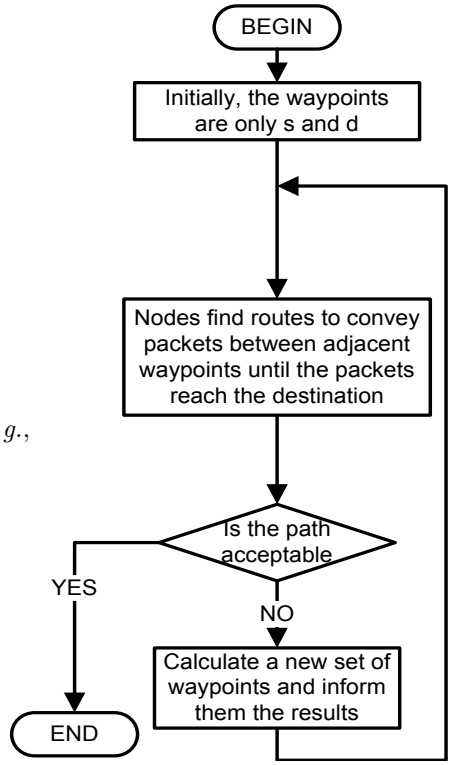
4.3.3 Overview of GDRP

In order to address the above two issues, we specifically design GDRP, a waypoint-based geographic data reporting protocol for WSNs. GDRP employs a trial-and-error approach to find an optimal path. Round by round, based on its current knowledge on the holes and barriers between the source and destination, it tries a so-far-the-best waypoint sequence to bypass the holes and barriers. Gradually it reinforces its knowledge on the holes and barriers, until eventually an optimal path is found to bypass the holes and barriers. Algorithm 4 illustrates its major mechanism.

Let j denote the round index. P_j denotes a path from the source to the destination found in round j and L_j denotes all the paths that have been found in round j and before, *i.e.*, $L_j = P_1 \cup P_2 \cup ... \cup P_j$. In the first round, since no knowledge on the in-network holes and barriers is available, the waypoints are only

Algorithm 4 The mechanism of GDRP

1: $EndFlag \Leftarrow false$
2: $L_0 \Leftarrow \emptyset$
3: $j \Leftarrow 0$
4: $W_1 \Leftarrow [s, d]$
5: **repeat**
6: $\quad j \Leftarrow j + 1$
7: $\quad P_j \Leftarrow \emptyset$
8: $\quad k \Leftarrow |W_j|$
9: $\quad i \Leftarrow k$
10: $\quad$ **while** $i \neq 1$ **do**
11: $\quad\quad P' \Leftarrow \text{route}(W_j(i-1), W_j(i))$
12: $\quad\quad P_j \Leftarrow P' \oplus P_j$
$\quad\quad$ /* $\oplus$ denotes the concatenation operation of two path segments, *e.g.*, $[a, b, c] \oplus [c, d] = [a, b, c, d]$*/
13: $\quad\quad i \Leftarrow i - 1$
14: $\quad$ **end while**
15: $\quad$ **if** accept(P_j) **then**
16: $\quad\quad EndFlag \Leftarrow true$
17: $\quad$ **else**
18: $\quad\quad L_j \Leftarrow L_{j-1} \cup P_j$
19: $\quad\quad W_{j+1} \Leftarrow \text{cal\_waypoint}(L_j)$
20: $\quad$ **end if**
21: **until** $EndFlag$ is *true*



the source and the destination. The path discovery scheme of GDRP finds the first path P_1 from the source to the destination. In round j ($j > 1$), the waypoint calculation scheme selects a sequence of waypoints W_j=$[w_j(1), \ldots, w_j(M_j)]$ based on L_{j-1}. And the path discovery scheme forwards packets between each adjacent waypoint pair $w_j(i-1)$ and $w_j(i)$ ($\forall i = 2, ..., M_j$) so that packets are conveyed from the source to the destination. A new path P_j is thus found. The above procedure is conducted iteratively until the resulting path is *acceptable*.

GDRP adopts a tidy framework where there are essentially only three components:

- Waypoint calculation, denoted by *cal_waypoint*$(\cdot)$ in line 19 of Algorithm 4;
- Path discovery between adjacent waypoints, denoted by *route*$(\cdot, \cdot)$ of line 11 in Algorithm 4;
- Examination of whether an existing path is acceptable, denoted by *accept*$(\cdot)$ in line 15 of Algorithm 4.

The first and the third components are conducted by only the sink, while the in-network nodes are responsible for the second component. The sink (*e.g.*, a laptop computer, a PDA, or a robot in a mobility-assisted WSN) is usually with more powerful computational capability and less energy constraint than the in-network sensor nodes. We hence put the major computational loads in the first and the third components. After each round of Algorithm 4, GDRP employs an intelligent waypoint calculation scheme (the first component) to find a new waypoint sequence for the next round, via which GDRP can potentially bypass the in-network holes and barriers. GDRP stops the path finding procedure in Algorithm 4 when a path satisfies a condition which can guarantee the performance of the path in

terms of topological length (the third component). Finally, the second component is carefully tailored for energy-constraint sensor nodes. It is based on the traditional geographic forwarding scheme, which is inherently light-weighted. These components will be illustrated in Section 4.4.

4.3.4 GDRP Preliminaries

Packet format

The header of a GDRP packet contains three fields: the geographic location of the intended destination, the location of the waypoint that the packet is currently heading for, and the locations of nodes the packet has visited. The packet format is shown in Figure 4.3.

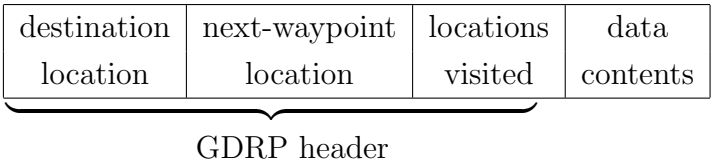


Figure 4.3: GDRP packet format

The *destination location* field saves the sink location, which is unchanged during the packet's trip to the sink. The *locations visited* field is incrementally updated by each node that forwards the packet. A record of such locations is generally required by a waypoint-based geographic forwarding approach for calculating

a waypoint sequence [8, 51, 124]. In round 1 of Algorithm 4, as only the source and the sink are waypoints, the source fills the location of the sink into the *next-waypoint location* field, meaning that the packet is heading for the sink. Then, a new waypoint sequence can be calculated in each round j ($j > 1$). When a packet reaches each designated waypoint, this field can be updated by the waypoint to its next waypoint.

Note that such a packet header is small in size. Most of the overhead is in the *locations visited* field, which, however, is only applied before Algorithm 4 converges. After GDRP finds a waypoint sequence that results in an acceptable path, such a field is then unnecessary. Hence, the packet overhead caused by the header is inconsiderable, which sticks to our objective of energy efficiency.

Waypoint table

For a node selected as waypoint, the sink would inform it and let it know the location of its next adjacent waypoint. A *waypoint table* is designed to save such information. Each record of the waypoint table contains only two fields: The *destination location* field and the *next waypoint location* field. The *next waypoint location* field tells where the next waypoint is for those packets

targeting at what the corresponding *destination location* field indicates. Finally, note that the waypoint tables are quite small in size (tens of bytes for typical WSNs), which is acceptable for the state-of-the-art sensor platforms [21].

4.4 Surviving Holes and Barriers with GDRP

4.4.1 When a Path is Acceptable

Let us firstly formulate *perfect sequence in geographic forwarding* (in short, perfect sequence) and *strongly* perfect sequence as follows.

Definition 3 *A sequence of nodes* $[u_0, u_1, ..., u_n, w]$ *is a perfect sequence in geographic forwarding if it satisfies:*

- *The distance between any two adjacent nodes in the subsequence* $[u_0, ..., u_n]$ *is less than or equal to* r*;*
- u_i *is geographically closer to* w *than node* u_k *if* $i > k$*.*
- *The distance between* u_i $(i > 0)$ *and any other nodes except* u_{i-1} *and* u_{i+1} *in* $[u_1, ..., u_n]$ *is larger than* r*;*
- *Given an* x*-*y *coordinate system with its* x*-axis passing* u_0 *and* w*, the maximum difference of the* y*-coordinates between*

any two nodes are no more than $d = \alpha \cdot r$, where α is a constant.

A perfect sequence is a strongly perfect sequence if:

- *The distance between w and u_n is less than or equal to r, while the distance between w and any other nodes in the sequence is larger than r.*□

The first three requirements guarantee that the nodes except the last in a perfect sequence form a *path segment* with the last node being the destination, where greedy forwarding is always feasible. A strongly perfect sequence further ensures all nodes form a greedy forwarding *path* towards the last node in the sequence. Moreover, the nodes in a perfect sequence are confined in a rectangular area based on the fourth requirement. As demonstrated in Figure 4.4, the whole path [u_0, u_1, ..., u_6, destination] is a strongly perfect sequence, while [u_0, u_1, u_2, u_3, destination] is a perfect sequence since the distance between u_3 and destination is larger than r. A good property of strongly perfect sequence is as follows.

Lemma 2 *The topological length of a path is bounded by $\frac{8(\alpha+1)}{\pi r}$ times the Euclidean distance between the source and the destination if the path is a strongly perfect sequence.*□

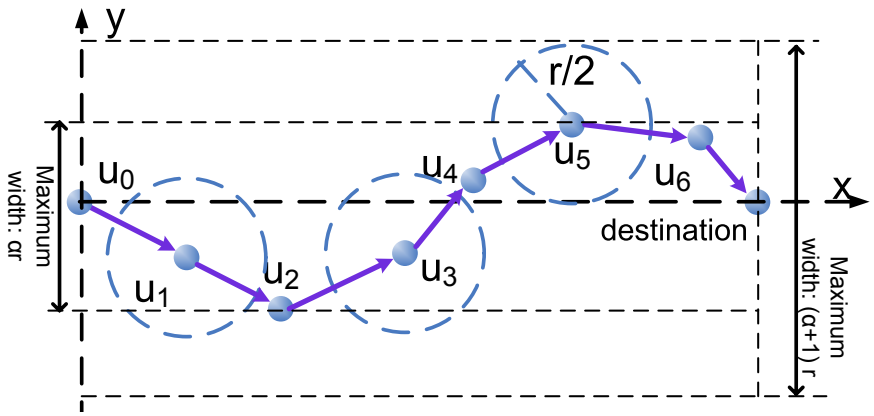


Figure 4.4: A strongly perfect sequence

Proof: Let us denote the path as [u_0, u_1, ..., destination]. Suppose the distant between the source to the destination is l. As shown in Figure 4.4, draw the circles centered at u_i where i is odd with their radii equal to $\frac{r}{2}$. All these circles must be in a rectangular area with length l and width $(\alpha+1)r$ because all nodes are in a rectangular area with length l and width αr. Moreover, these circles should not intersect each other. Otherwise, the distance between two non-adjacent nodes is less than r which contradicts the requirements of strongly perfect sequence. Therefore, the maximum number of such circles $n(l)$ are bounded by:

$$n(l) \leq \frac{l \cdot (\alpha+1)r}{\pi(\frac{1}{2}r)^2} = \frac{4(\alpha+1)}{\pi r} \cdot l. \tag{4.1}$$

The total number of hops (*i.e.*, the number of node in the sequence minus one) in the path is bounded by twice of such a

circle number. The topological length of a strongly perfect sequence, denoted by $H_{SPS}(l)$, is hence bounded by:

$$H_{SPS}(l) \leq \frac{8(\alpha+1)}{\pi r} \cdot l, \tag{4.2}$$

which is linearly related to l and thus Lemma 2 is proved.

Corollary 1 *If a path between two nodes is a strongly perfect sequence, the topological length of the path is not larger than* $\frac{8(\alpha+1)}{\pi}$ *times that of their shortest path.*

Proof: Let us again suppose the distance between the source and the destination is l. The topological length of the shortest path $H_S(l)$ between them is lower-bounded by $\lceil \frac{l}{r} \rceil$. Therefore according to Lemma 2, we get:

$$H_{SPS}(l) \leq \frac{8(\alpha+1)}{\pi} \lceil \frac{l}{r} \rceil \leq \frac{8(\alpha+1)}{\pi} H_{SP}(l). \tag{4.3}$$

It shows that the topological length of a strongly perfect sequence will not be worse than $\frac{8(\alpha+1)}{\pi}$ times of that of the shortest path, which proves Corollary 1.

Therefore, we deem that given a path segment from waypoint $w(i\text{-}1)$ to $w(i)$, if the path segment is a strongly perfect sequence, the in-network holes and barriers do not considerably influence the data transportation between $w(i\text{-}1)$ and $w(i)$. Hence, GDRP

can accept a path and stop Algorithm 4 if the path is an acceptable path defined as follows.

Definition 4 *A path P_j is an acceptable path for Algorithm 4 if the path segments between any two adjacent waypoints, i.e., $w_j(i\text{-}1)$ and $w_j(i)$ ($\forall i = 2, ..., M$), are strongly perfect sequences.*□

4.4.2 Calculating a New Waypoint Sequence

In order to effectively bypass holes and barriers, the waypoints should be determined based on the locations of the holes and barriers. However, since the sink does not have a global picture of the network, holes and barriers cannot be known beforehand. GDRP has to learn the information of the holes and barriers with the paths found in previous rounds. With a trial-and-error approach, the knowledge on the holes and barriers can be accumulated. Consequently, better and better waypoint sequences can be designed. In what follows, we first discuss how GDRP learns the knowledge of the holes and barriers with the paths found in previous rounds. And then, we provide how GDRP calculates a waypoint sequence based on such knowledge.

Modeling the influences of holes and barriers

When a path is not an acceptable path, there exists at least one path segment between a pair of adjacent waypoints which is not a strongly perfect sequence. Let $[w, u_1, u_2, ..., w']$ denote such a path segment where u_1, ..., and u_i are the intermediate nodes, and w, w' are the adjacent waypoints. In this case, there are some holes or barriers that influence the packet transportation from w to w'.

Since the impact of holes or barriers can be considered as how they make the path segment "imperfect", they can be inferred by finding which parts of the segment make it fail to be a strongly perfect sequence. Thus, we can model the impact of the holes and barriers using these parts. Figure 4.5 demonstrates an example. In this example, $[w, u_1, u_2, w']$, $[u_4, u_5, u_6, w']$, and $[u_8, u_9, u_{10}, w']$ are all perfect sequences; whereas $[u_2, u_3, u_4]$ and $[u_6,\ u_7,\ u_8]$ make the whole path segment fail to be a strongly perfect sequence. Therefore, it can be inferred that there are some holes or barriers lying in the shaded areas shown in Figure 4.5.

Specifically, for $[w, u_1, u_2, ..., w']$, GDRP first finds those parts that form perfect sequences together with w'. And then it considers the rest parts are resulting from the holes or barriers be-

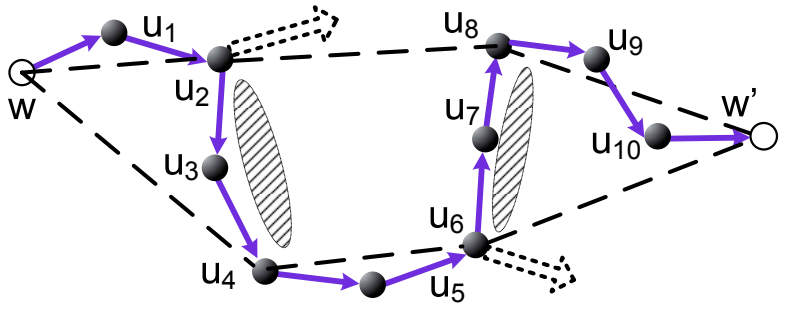


Figure 4.5: An example on modeling the influences of holes/barriers and calculating waypoint sequence

tween the two waypoints. We call such a part a *detour part*, where the first node is called the *starting node* of the detour part and the last one is its *ending node*. Consider the example case shown in Figure 4.5. $[u_2, u_3, u_4]$ and $[u_6, u_7, u_8]$ are two detour parts with starting nodes being u_2 and u_6, and ending nodes being u_4 and u_8, respectively.

Thus, for each path P_j found in round j of Algorithm 4, if the path is not an acceptable path, a set of detour parts in P_j, denoted by B_j, can be found. And then the collection $\{B_k\}_{k=1}^{j}$ can be regarded as the current knowledge of the in-network holes and barriers by the end of round j of Algorithm 4. Note that since the impacts of the holes and barriers are the same based on the notion that they both make a path contain detour parts, formulating their impacts as the detour parts actually provide GDRP a generic treatment to both holes and barriers.

Obtaining the best waypoint sequence

Based on the current knowledge $\{B_k\}_{k=1}^{j}$, our objective is to find a best set of waypoints W_{j+1} for round j+1 so as to make the packets bypass the known holes and barriers in round j+1 and minimize the topological length of the path found in round j+1.

We consider the starting nodes and the ending nodes of the detour parts as the *potential waypoints*, *i.e.*, the waypoints in W_{j+1} are selected from the set of the starting nodes and the ending nodes of all detour parts in $\{B_k\}_{k=1}^{j}$. In the example shown in Figure 4.5, u_2, u_4, u_6, and u_8 are hence the potential waypoints. The reason is justified as follows.

We select an ending node of a detour part as a potential waypoint because from this node on the path is clear of the hole or barrier modeled by the detour part, since a new perfect sequence starts from this node. In other words, the known hole or barrier modeled by a detour part does not influence the path any longer from its ending node on.

We select a starting node as potential waypoint as we take into accounts the possibility that the node can avoid the corresponding detour part by just forwarding packets to another direction. Again consider the example in Figure 4.5. If u_2 and u_6 are waypoints, they can attempt to explore the network in the

other directions indicated by the dotted arrows in the figure to avoid the known holes or barriers modeled by their corresponding detour parts. Finally, note that when a node is the starting node of more than one detour part, it means that both directions have been tried. Consequently, it would not be considered as a potential waypoint.

Given the potential waypoints composed by the starting nodes and the ending nodes of $\{B_k\}_{k=1}^{j}$, we expect in round j+1 GDRP can find an acceptable path, *i.e.*, GDRP can find a strongly perfect sequence between each pair of the adjacent waypoints. Therefore, draw a line segment between two adjacent waypoints in W_{j+1}, it cannot intersects a known detour part. Otherwise, the path segment found in round j+1 between the waypoints cannot be a strongly perfect sequence as it will detour when reaching the hole or obstacle modeled by the detour part. Based on this consideration, we calculate the waypoint sequence W_{j+1} as follows.

First, construct a subgraph G_W of the network in which the vertex set includes the potential waypoints, the source s, and the sink d. Two vertices in G_W share an edge if and only if the line segment between the two vertices does not intersect any

of the detour parts[8]. Figure 4.5 shows an example where the dotted line segments denote the edges of G_W. The waypoint sequence W_{j+1} should form a path in G_W since we expect that path segments between any two adjacent waypoints will not intersect a known detour part. We then let each edge in graph G_W be weighted by its Euclidean distance and find a shortest path in G_W from the source s and the destination d. The potential waypoints along such a shortest path is hence considered as the resulting waypoint sequence.

In the example shown in Figure 4.5, the shortest path is $w \rightarrow u_2 \rightarrow u_8 \rightarrow w'$ and hence the waypoint sequence is [w, u_2, u_8, w']. Lastly, note that no additional overhead is introduced in this procedure since G_W is constructed merely with L_j=$\{P_k\}_{k=1}^{j}$ collected in each round of Algorithm 4. Based on how G_W is constructed and the property of shortest path, we can obtain the following lemma.

Lemma 3 *The waypoint sequence found by GDRP is with minimum Euclidean length according to the current knowledge of the holes and barriers.*

Suppose in round k, GDRP eventually converges. The topological length of P_k is then linearly related to the Euclidean

[8]The starting node and the ending node of the same detour part will not share an edge.

length of W_k. Since the Euclidean length of W_k is the minimum, we can hence guarantee the resulting path is an optimal path between the source and the destination.

4.4.3 Geographic Forwarding between Adjacent Waypoints

Now we discuss how we tailor the path discovery scheme of a sensor node so that it can conduct the geographic forwarding task between two adjacent waypoints. First of all, such a path discovery scheme should always ensure successful packet delivery between two adjacent waypoints. Hence we base it on the path discovery scheme adopted in traditional geographic forwarding approaches (see Section 4.3.1) [56]. But two changes are made as follows.

The first change is that the right-hand rule or left-hand rule may be adopted even if greedy forwarding is feasible. This is due to the fact that even greedy forwarding is feasible, there may be a hole or barrier ahead. GDRP should try to forward packet to another direction to bypass the hole or barrier.

The second change is that in GDRP, the staring node of a detour part (*i.e.* a node which faces a hole or barrier ahead) can choose the opposite direction to what is chosen in the previous

round of Algorithm 4, so as to explore a new path which is potentially better than what is found in the previous round. For conducting this task, we design a *direction table* for each starting node. Each record of a direction table contains two field: the *waypoint* field and the *direction* field. The *direction* field saves the rule (left-hand or right-hand) adopted in the previous round of Algorithm 4 when forwarding packets to the waypoint recorded in the corresponding *waypoint* field. After each round of Algorithm 4, if GDRP finds out that a node is a starting node of an newly-found detour part, it would inform the node. Suppose in the previous round the packets forwarded by this node is heading for a waypoint w. The node would then check its routing direction table and find the record of w and change the direction in its *direction* field accordingly[9].

We now elaborate the behavior of a node when it runs GDRP. First, when receiving a packet, a node reads the packet's *next-waypoint location* field. If the node is the packet's intended waypoint, it would update the *next-waypoint location* field of the packet according to the node's waypoint table before it sends the packet. Otherwise, it just proceeds to send the packet.

When sending a packet, if no record of the corresponding

[9]If there is no record of w, the node would create one for w. The location of w is saved in its *waypoint* field. The *direction* field is set to the opposite direction of the default one.

waypoint can be found in a node's direction table or there is no direction table, the node would simply adopts the same strategy as that in traditional geographic forwarding to send the packet: Employ greedy forwarding if it is feasible, or otherwise take predetermined left-hand rule or right-hand rule and enter the detour mode to forward the packet. But if the record can be found, it will choose a routing direction according to the *direction* field of the record and adopt left-hand rule or right-hand rule accordingly .

4.4.4 A Case Demonstration of GDRP

Figure 4.6 demonstrates the resulting paths found in each round of GDRP in an example network. In the first round, since no network holes or barriers are known, the sink considers [source, sink] as the waypoint sequence. The path in Figure 4.6(a) is found. the sink finds two potential waypoints w_1 and w_2 shown in the figure. It then informs w_1 to change its forwarding direction since it is the starting node of the detour part from w_1 to w_2.

Because the line segment from the source to the sink does not intersect the known detour part from w_1 to w_2, in round 2, the sink again sets [source, sink] as the waypoint sequence

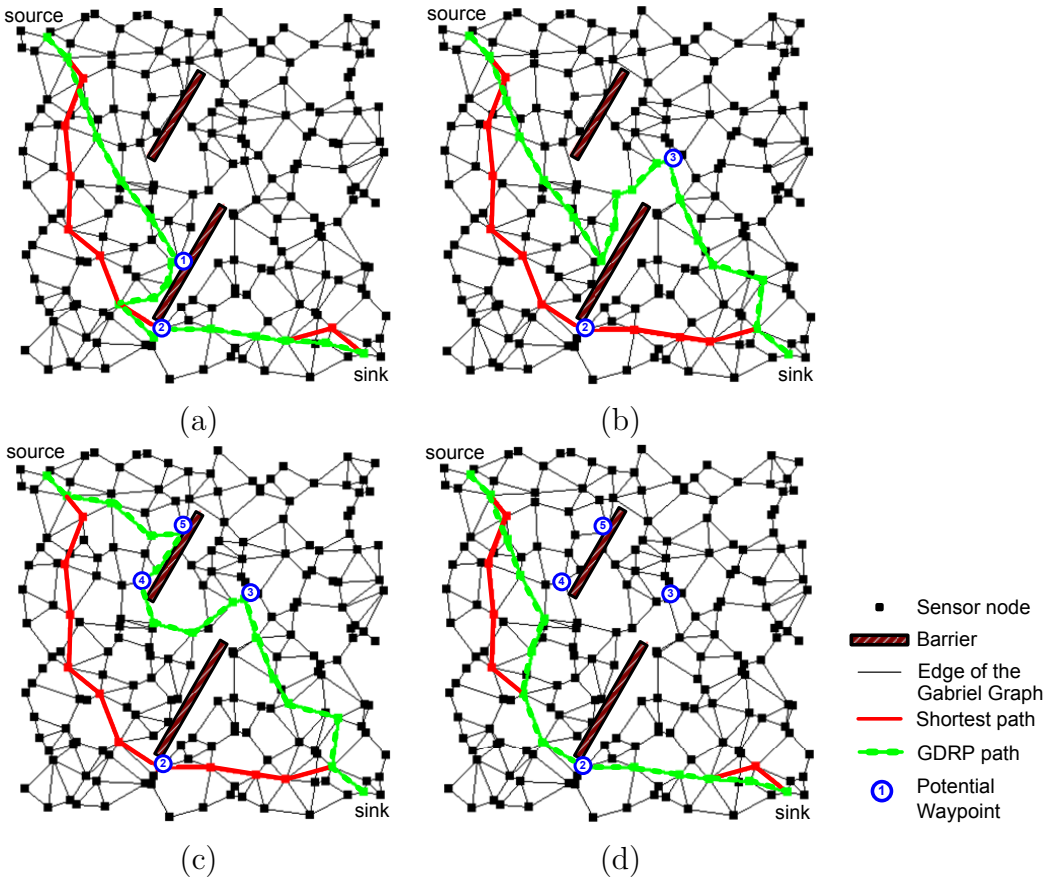


Figure 4.6: A case demonstration of GDRP

since it is the shortest path in G_W from the source to the sink. Now when a packet reaches w_1, w_1 would forward the packet to anther direction. The resulting path is shown in Figure 4.6(b). Another detour part from w_1 to w_3 is found and w_3 is then a potential waypoint. Since w_1 is the starting node of two detour parts, the sink knows w_1 fails to bypass a hole or barrier and hence w_1 will not be selected as a potential waypoint any more.

Now the sink finds out that the waypoint sequence [source,

w_3, sink] is the shortest path in G_W from the source to the sink. It selects w_3 as a waypoint and informs it in round 3. Packets are now first sent to w_3 and then to the sink. Figure 4.6(c) shows the resulting path. The sink finds the third detour part from w_5 to w_4. w_4 and w_5 are then the new potential waypoints and G_W is updated.

Now the waypoint sequence [source, w_2, sink] is the shortest path in G_W from the source to the sink. In round 4, the sink sets w_2 as a waypoint and informs it. In this round, the path segments from the source to w_2 and from w_2 to the sink are both strongly perfect sequences as shown in Figure 4.6(d). GDRP completes its task in finding a best set of waypoint. Packets, from then on, can be forwarding along the path found in round 4.

For comparison purpose, we also plot the shortest path in Figure 4.6. It shows the final resulting path of GDRP is comparable to the shortest path in terms of the topological length.

4.5 Simulation Study

In order to study the effectiveness of GDRP in surviving network holes and barriers, we simulate a WSN. Energy-efficiency is studied in terms of the topological length of the resulting paths. We

compare our GDRP protocol with GPSR (a geographic forwarding protocol proposed in [56]) and a waypoint-based geographic forwarding protocol like that proposed in [51], which is named CONVEX-W in this thesis. GPSR tolerates holes and barriers by entering the detour mode instead of relying on waypoints. CONVEX-W always find waypoints in one side of the line segment from the source to the sink, which forms the half convex hull of the source, the sink, and the known holes and barriers in between. In our simulations, we also find shortest paths with global network information for comparison purpose.

All the geographic forwarding protocols planarize the network based on its Gabrial Graph for the detour mode in our simulation studies. The default forwarding direction in the detour mode takes the right-hand rule. The constant α for GDRP in determining perfect sequences is set to 2 empirically. the sink is located at one corner of the network area ($\frac{r}{2}$ away from two boundaries), while the source is located at its opposite corner (also $\frac{r}{2}$ away from two boundaries). Network holes and barriers are simulated by inserting ellipses and line segments into the network. The line segments cut the intersecting wireless links, while the ellipses are the areas in which there are no sensor nodes. The ellipses and the line segments do not intersect the

network boundary to avoid geographic forwarding routes packets along network perimeters. Finally, the details of the simulation network settings are shown in Table 4.1, which are typical WSN settings. For each setting in our following performance studies, we adopt 60 different random seeds in every runs and the results are averaged. We do not consider the cases that geographic forwarding fails to deliver a packet, which is generally caused by the inserted holes and barriers that result in an unconnected network.

Table 4.1: Simulation settings in Chapter 4

Area of sensor field	$400m \times 400m$
Node deployment scheme	Randomly deployed in a uniform manner
Node communication range r	$[40m, 70m]$
Sensor node number	$[100, 500]$
Holes and barrier number	$[2, 8]$

We first study how the number of network holes and barriers influences the topological lengths of the paths. Figure 4.7 demonstrates the results where the communication range is $60m$ and the number of sensor nodes is 500. It shows that as the number of holes and barriers increases, the paths found by all the protocols result in larger topological length. This is because with more holes and barriers in between, greedy forwarding faces

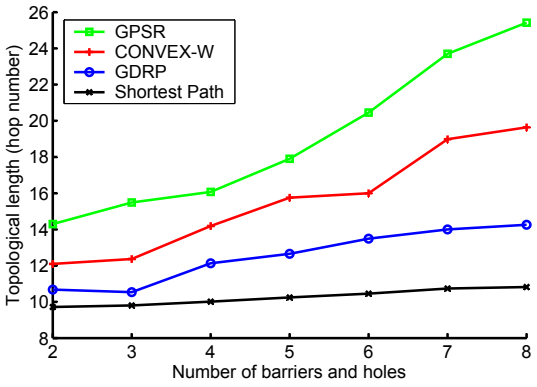


Figure 4.7: Topological lengths with different numbers of holes and barriers

higher chances to fail, which results in longer paths for bypassing more holes and barriers. We can see that our GDRP always outperforms GPSR and CONVEX-W. This verifies the advantage of the waypoint selection algorithm adopted in GDRP. Note that even when the hole and barrier number is large, unlike that of the other two protocols, the performance of GDRP does not dramatically deviate from the optimum shortest path. This shows the effectiveness of GDRP in surviving network holes and barriers.

We also study how GDRP performs with respect to the node density and node communication range. Six holes and barriers are injected into the network. Figure 4.8 shows the results where we let the communication range be $60m$ and change the node number from 100 to 500, and Figure 4.9 provides the results where we set the node number to 400 and change the commu-

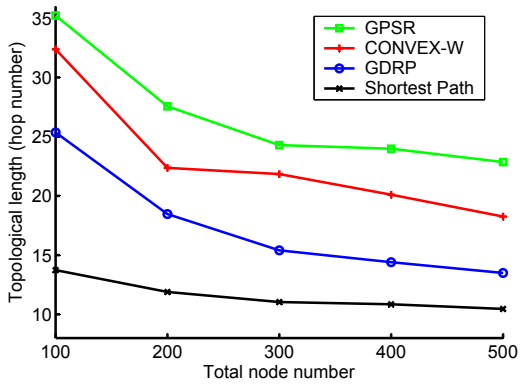


Figure 4.8: Topological lengths as a function of in-network node number

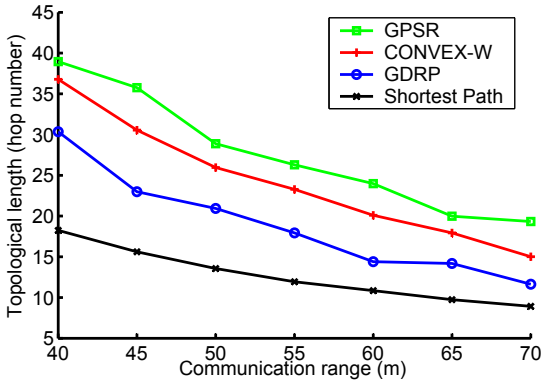


Figure 4.9: Topological lengths as a function of communication range

nication range from 30 to 60.

It can be seen that the larger the node density or communication range is, the better all these protocols perform. This is quite natural: Larger node density or communication range result in larger per-hop progress in geographic forwarding [138]. Moreover, larger node density or communication range also decreases the chance that greedy forwarding fails since more neighbors are available as the potential greedy forwarding candidates. These

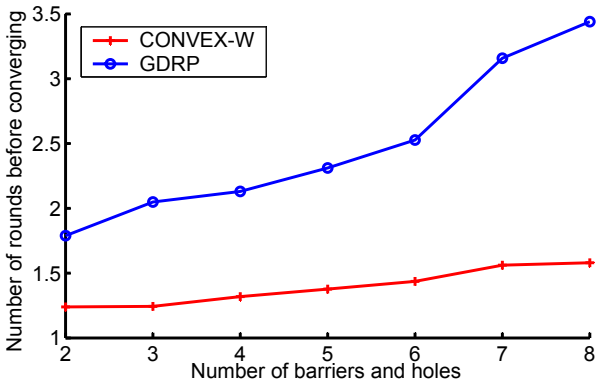


Figure 4.10: Average number of rounds needed before converging

results further confirms that GDRP always outperforms GPSR and CONVEX-W in different network settings.

Finally, although GDRP performs well in terms of the topological lengths of the paths it finds, an important concern is the number of rounds it requires in finding an acceptable path. Figure 4.10 compares GDRP and CONVEX-W since CONVEX-W is also an iterative protocol. The communication range is $60m$ and the number of sensor nodes is 500 in this study. It shows that although GDRP averagely needs more rounds to converge than CONVEX-W does, the round number is still small even when the hole and barrier number is high. Moreover, it do not grow dramatically with the hole and barrier number. This shows the nice tractability of the waypoint selection problem modeled in GDRP.

4.6 Conclusion

Geographic forwarding has long been advocated as a promising technique in transporting sensor data in WSNs. However, in practical WSN deployments, network holes and barriers are inevitable. This poses a critical challenge to traditional geographic forwarding and consequently makes it very inefficient in terms of energy consumption. In this chapter we aim to improve the energy efficiency of geographic forwarding, so as to enhance the survivability of the network in practical deployment environments. We address this problem by proposing a waypoint-based geographic forwarding approach called GDRP. We prove the performance guarantee of GDRP and verify its effectiveness in tolerating network holes and barriers with extensive simulation studies.

□ **End of chapter.**

Chapter 5

Energy-Efficient On-demand Active Contour Service

Summary

Contour mapping is an important technique for WSNs to abstract the information of a monitored field. State-of-the-art approaches for contour mapping, however, are neither energy-optimized, nor capable of handling heterogeneous user requests. We hence develop a novel On-demand Active Contour Service (OACS) for WSNs. OACS regresses the field intensity function with kernel Support Vector Regression, which flexibly handles both contour line and contour map requests. OACS also adaptively accommodates a wide range of contour line/map precision requirements: (1) For applications of low precision, only a minimum set of nodes are scheduled in the on-duty mode while others are sleeping for conserving energy. (2) For applications of high precision, via an active and progressive learning algorithm, OACS determines the best set of nodes that should be turned on for improving the contour line/map precision. Evaluation based on diverse realistic models demonstrates that OACS provides quality and seamless contour services for various application requirements yet significantly conserves energy.

5.1 Introduction

In data collection WSN applications, the major task of a WSN is to measure, process, and convey the sensor data to a data sink so that the sink can reconstruct the information of a scalar field of interest, *e.g.*, the temperature distribution throughout a monitored space, or the boundary where the concentration of a toxic gas reaches a dangerous level. Since such a field is continuous in nature, *contour mapping* turns out to be a natural and necessary service: Based on the in-situ sensor readings, the sinks of WSNs can construct contour lines or contour maps to present the information of the scalar fields. A *contour line* is a line along which the intensity of the field is a constant value. The value is hence called *the value of the contour line*. A *contour map* is a map illustrated with a set of such contour lines.

Contour mapping has long been recognized as an important approach for WSN applications [75], *e.g.*, to locate and monitor a phenomenon of interest [119] and to capture the WSN system life conditions such as the energy levels of the nodes [125]. It can also provide instant and user-friendly visualization of the scalar field of interest [68, 118].

One challenge faced by contour mapping in WSNs is how to handle diverse contour requests from users. A user may request a single contour line to obtain the information of a boundary, or request a contour map to obtain the information of the entire field of interest. Moreover, the user may also have different requirements on the precision of a contour line/map. For example, he/she may be interested in observing more details of the field when a particular phenomenon takes place and hence may require the network to produce a finer map. However, a contour service that can handle such diverse user requests with energy-efficiency is still at large in the current state of the art.

The existing contour mapping approaches are generally optimized for either contour line (*e.g.*, [68, 98, 118]) or contour map (*e.g.*, [75, 95]) requests, but not for both. More importantly, they largely ignore the fact that the users may have different requirements on the precision of a contour line/map. Their focus is to minimize the number of packets that should be reported to the sink for achieving energy efficiency. They generally require each in-network node to sense the environmental data of interest, although the sensor readings are processed in the network and some are possibly suppressed. As a result, they are essentially *best-effort* approaches and lack a scheme for tuning the

precision of the line/map by controlling the number of the on-duty nodes [40, 68, 75, 98, 118]. Such precision tuning, however, is not just a marginal service consideration, but very crucial for energy saving. With such a scheme, we can put some in-network nodes into the sleeping mode and thus save more energy when low precision is acceptable.

To address this critical challenge, we develop a novel energy-efficient contour service, namely, On-demand Active Contour Service (OACS), for WSNs to seamlessly handle diverse user requests. OACS first divides the network area into clusters. It then regresses a *field intensity function* in each cluster by taking sensor readings with sensor locations as input samples. This provides the flexibility to handle contour line and contour map requests adaptively. For a contour line request, what we focus is to let the regression result accurately represent the reality in the area around the real contour line. Hence, only the sensor readings close to the contour line value are required for the regression. Whereas for contour map requests, since the contour lines of interest are spreading throughout the network area, the regression result is important everywhere of the network. Hence more sensor readings can be involved in the regression.

To deal with different requirements on contour line/map pre-

cision, OACS initially requires only a minimum set of nodes are on duty while putting the rest in the sleeping mode. According to the precision requirement, OACS then progressively activates sleeping nodes to enhance the precision expected by the users. To this end, it incorporates a comprehensive set of selection algorithms for the nodes-to-be-activated, which are enlightened by the recent advances in *active learning* [106]. This novel two-step design of OACS seamlessly accommodates request diversity yet significantly conserves energy.

The rest of this chapter is organized as follows. Section 5.2 studies the related work. In Section 5.3, we provide a system-level overview of our contour service. Section 5.4 analyzes the contour mapping problem and a two-step regression approach is illustrated. Section 5.5 reports the performance study results and Section 5.6 provides the conclusion remarks.

5.2 Related Work

Contour mapping has long been studied by geography and geology researchers [22, 29], where their concern was mainly on how to generate a contour map with all data at hand. A straightforward application to WSNs is that the sink collects all individual sensor readings and then builds the map in a centralized manner

(*e.g.*, with linear interpolation such as Kriging [59]). Such an approach, although simple, incurs too much energy consumption as every node has to sense and report data to the sink. It is not suitable WSNs due to their energy constraints. This leads to the design of many *distributed* approaches. Some proposed that only the nodes with readings close to the value of the contour lines report their readings to the sink (*e.g.*, [68, 98]). In other approaches, a network is divided into clusters. In each cluster, segments of contour lines are constructed at its cluster head and reported to the sink instead of all the sensor readings (*e.g.*, [118]).

The idea of using machine learning techniques in contour mapping was recently suggested in [118]. It formulates the problem of constructing a segment of a contour line as a non-linear classification problem. A severe algorithmic inadequacy of this approach is that it relies heavily on the shape of the distribution of the nodes' locations, while disregarding their sensor readings. As a result, it may generate biased contour lines and generate the same curve for the contour lines with different values. In addition, the application of linear regression to contour mapping was proposed in [68]. But a contour line is non-linear in nature. Such a linear approximation generates results that are

either inaccurate or with too many parameters. Moreover, a mobility-assisted approach was proposed in [99], which studies how to design the tracks of a set of mobile nodes to detect a contour line. The problem context, however, is quite different from that studied in this chapter.

Note that the above schemes are essentially optimized for calculating a single contour line. When an entire contour map is required, they usually need to construct each contour line separately and eventually the resulting lines are combined into one map [68, 98, 118]. This is actually very inefficient. Neighboring contour lines usually exhibit space correlations, and hence it is possible to construct and represent them in a better way. Taking this into account, many approaches try to exploit such spatial correlations in building an entire map. The idea to abstract a field with isobars (*i.e.*, rectangle area where the field intensity is close to a value) was suggested in [80] for boundary detection. Hellerstein *et al.* further proposed to build such an isobar map based on a routing tree [48]. Gandhi *et al.* suggested summarizing a contour map with a topologically equivalent family of polygons [40]. Silberstein *et al.* [95] and Meng *et al.* [75] investigated algorithms for spatial and temporal suppression. Their focus is a data aggregation scheme for minimizing en-

ergy consumption of communication. Similarly for efficient data management, Guestrin *et al.* proposed to compress spatiotemporally correlated sensor data with a distributed regression approach [44]. These approaches, however, are not efficient for calculating a single contour line since they aim at abstracting *all* in-network sensor readings.

To the best of our knowledge, no comprehensive yet adaptive treatment to both contour line and contour map requests is presented in the literature. Moreover, the existing approaches are best-effort only: There is no mechanism for energy conserving when lower precision contour line/map is acceptable. Finally, they lack a proper scheme to actively select sensor nodes in calculating contour lines/maps.

5.3 System-level Overview of On-demand Active Contour Service

This section provides an overview of the OACS approach in a system perspective. We first identify its working environments. Then we discuss the user requirements and how we encode them in OACS's user enquiries. Finally, we brief OACS's major work flow in processing its user enquiries.

5.3.1 Preliminaries

We consider a WSN consisted of N stationary sensor nodes $\{s_j\}_{j=1}^{N}$ randomly deployed in a 2-dimensional network area ϕ (*i.e.*, $\phi \subset \mathbb{R}^2$). Like work in [118], we consider the network consists of K clusters and there is a head h_i for each cluster i [18, 123]. Cluster heads are in charge of constructing the segments of contour lines/maps inside their clusters and reporting the results to the sink. They are generally equipped with a high computational capability hardware platform, *e.g.*, the Crossbow Imote2 [19]. Cluster members, on the other hand, can be relatively low-capability nodes such as the Crossbow IRIS Mote [20]. Let $\mathcal{S}_i$ denote the set of the nodes in cluster i, *i.e.*, s_j is in cluster i if and only if $j \in \mathcal{S}_i$. n_i is the node number in $\mathcal{S}_i$. L_j denotes the location of s_j. Location-awareness is a basic requirement for contour mapping approaches [40, 68, 75, 98, 118, 119]. Hence, we consider that each node is aware of its own location, which can be obtained by a light-weighted localization approach (*e.g.*, Calamari [114] distributed with TinyOS [105]). Cluster heads thus know the locations of their cluster members.

Suppose among all nodes in a cluster, there are a portion of *on-duty nodes*, *i.e.*, those that are not in the sleeping mode,

which maintain the normal sensing task of the network[10]. Let λ denote the percentage of the on-duty nodes in a cluster. We assume that there is a mechanism for cluster heads to let any of its sleeping cluster members change to the on-duty mode. This can be done via an active scheme like that proposed in [43] with which a node can be turned on by radio, or a passive scheme in which a sleeping node periodically turns itself on and enquires whether it should change to the on-duty mode.

z_j denotes the sensor reading of node s_j when s_j is on duty. We consider that $z_j \in \mathbb{R}$, *i.e.*, z_j is a 1-dimensional scalar value, *i.e.*, the measured intensity of *one* scalar field. This is assumed without loss of generality. If there are multiple co-existing fields, z_j is then multi-dimensional, resulting in multiple contour maps, each corresponding to one dimension. Each map can be constructed in the same way as building a map for a 1-dimensional z_j.

5.3.2 Contour Enquiries

We consider two natural user requirements of contour mapping service. The first one is that a user wants to obtain one sin-

[10]The roles of being on duty and being sleeping can be rotated for all in-network nodes based on a node grouping mechanism (*e.g.*, [67, 128]) to balance the residual energy of the nodes or to maintain coverage.

gle contour line with a given value. Identifying an phenomenon boundary (*e.g.*, the boundary of a pollution area as considered in [99]) is an example. The second one is that the user requires a contour map of the whole network area, for example, for visualization of the information of the entire field of interest. Hence, user enquiries are divided into two types, namely, line-enquiries and map-enquiries (*L-enquiries* and *M-enquiries*). In an L-enquiry, a user should give the value of the requested contour line. While in an M-enquiries, since the user requests the network to abstract the *entire* field information, no specific values are required.

Another important issue is the precision of the contour line or map a user requests. During a WSN system run-time, a user may require contour lines/maps with different precisions. For example, a user may ask the system to generate some coarse-grained contour maps periodically for logging the changes of a field. Whereas during a particular time interval (*e.g.*, when a phenomenon of interest takes place), the user may need a finer map for more details of the field.

Since neither the *actual* field intensity nor even its distribution is *a priori* knowledge, it is impossible to adopt traditional precision metrics such as mean square error (MSE) for present-

ing the precision of a calculated contour line/map: For example, if a user needs a contour line with its value being 10 and MSE being 0.5. After calculating a contour line with sensor readings, no one can tell whether its MSE satisfied the requirement or not since there is no way to know the ground-truth contour line. This poses a challenging problem: The precision requirement of an enquiry has to be defined in a way that it should not only make sense of how well a contour line/map matches the reality but also be a calculable value.

Note that the most precise contour line/map segment a cluster head h_i can get is the one constructed based on the readings of all nodes in $\mathcal{S}_i$. A contour service can generally provide a more precise result if it takes the readings of more nodes into account in constructing a contour line/map. Hence, the number of the involved nodes is naturally a good index to the precision of the result. The more the number of nodes involved is, the more precise the result is, whereas the more energy is required since more nodes are needed to sense and report their readings. Tuning the number of the involved nodes provides the flexibility for a user to trade off energy and the precision of the result.

Hence, the formats of enquiries are as follows. An L-enquiry is a 2-tuple $L{=}[cv, p]$ and an M-enquiry is denoted by $M{=}[p]$,

where cv is the value of the requested contour line, and p is the precision requirement, which is the percentage of the sensor node number in each cluster that the user requires to build a contour line/map.

5.3.3 Service Overview

Our contour service is an on-demand service: A user-enquiry can be directly broadcast to the sensor nodes by the sink with its powerful antenna, or can be broadcast through multi-hop. When receiving an M-enquiry, all on-duty nodes report their readings to their cluster heads, while when receiving an L-enquiry $[cv, p]$, only those on-duty nodes whose readings are close to the contour value cv report their readings to their cluster heads.

Each cluster head then knows the intensity of the field at a set of in-network locations. With these data, it computes the contour line/map segment inside its corresponding cluster with a kernel-based Support Vector Regression (SVR) technique.

It then examines whether the precision requirement of the results matches the user requirements p with the initial on-duty nodes. If the answer is *no*, it will employ an intelligent scheme to select a set of sleeping nodes actively, turn them on, obtain their readings, and refine the results of the kernel SVR scheme, until

the results are satisfactory. Finally, after the results are reported to the sink for combining into an overall contour line/map, the additional on-duty nodes go back to the sleeping mode.

5.4 Regression-based Contour Mapping Algorithms

Since the cluster heads play important roles in our contour service, in this section we focus on how a cluster head constructs the part of the contour line/map that is located in its cluster.

5.4.1 Contour Mapping with Kernel SVR

The intensity z of a scalar field is essentially an unknown continuous function $f(\mathbf{x})$ defined on ϕ, where $\mathbf{x}$ is a 2-dimensional vector denoting a location in the network area:

$$z = f(\mathbf{x}) \quad (\mathbf{x} \in \phi). \tag{5.1}$$

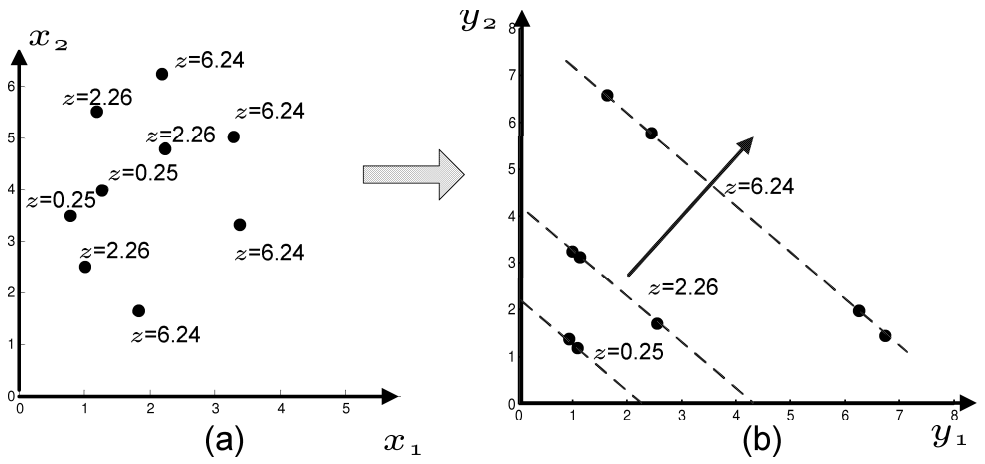
An actual contour line with value cv is then a curve where the points on the curve satisfy $f(\mathbf{x}) = cv$. While an entire contour map is composed by a lot of such curves (with values forming an arithmetic sequence) spreading throughout the network area. Given an estimation of the function $f(\mathbf{x})$, we can easily find out

a contour line/map.

The sensor readings can be deemed as the sample values of the function $f(\mathbf{x})$ where the nodes locate. Note that these sensor readings, however, are subject to measurement errors due to the influence of noise. Hence, contour mapping is essentially a procedure we regress $f(\mathbf{x})$ based on these noisy sample values.

Formally, for each cluster i, give its head h_i a set of samples, *i.e.*, the mappings $\{L_k \mapsto z_k\}_{\forall k \in \mathcal{W}_i}$ where $\mathcal{W}_i$ is a subset of all nodes in $\mathcal{S}_i$. Its goal is a function $\widetilde{f}_i(\mathbf{x})$ to estimate $f(\mathbf{x})$ in its cluster area, which satisfies $\widetilde{f}_i(L_k) = z_k + \epsilon_k$. Here ϵ_k is the error of sensor s_k. We assume the error $\{\epsilon_j\}_{\forall j \in \mathcal{S}_i}$ are independent and identically-distributed according to a zero-mean Gaussian distribution with variance σ^2, *i.e.* $\epsilon_j \sim N(0, \sigma^2)$.

To make this problem tractable, we need to assume a form of the function $\widetilde{f}_i(\mathbf{x})$. Work in [68] adopts a linear form to study this problem. Although linear functions are the simplest for this problem, they are not good candidates since the intensity of a field is generally non-linear in nature (and contour lines are generally not straight line segments). A better way is to use a combination of polynomial functions or Gaussian functions to represent $\widetilde{f}_i(\mathbf{x})$. This can be handled by kernel SVR [26]. We briefly present the approach as follows.



(a) Sensor readings are not linear to their locations. (b) Map sensor locations to another space, where sensor readings are linear to their locations.

Figure 5.1: An example of space mapping

Theoretical foundation

The idea of kernel SVR is employing space transform to map the original non-linear function $f(\mathbf{x})$ to another space where the function can be deemed as a linear function, which is then easy to be regressed. Figure 5.1 demonstrates a simple example of how space mapping works. In the network area, the sensor readings and their locations are not linearly-related. But, with a non-linear mapping y_1=x_1^2-2x_1+2 and y_2=x_2^2-8x_2+17, a location $\mathbf{x}$=(x_1, x_2) in the original space is then mapped to a location $\mathbf{y}$=(y_1, y_2) in another 2-dimensional space where we can regress z as a linear function of $\mathbf{y}$. For example, z=y_1+y_2-2 is a satisfactory function. The result can then be projected back to the original space, *i.e.*, z=$(x_1$-1$)^2$+$(x_2$-4$)^2$.

Let us suppose Φ is an $\mathbb{R}^2 \mapsto \mathbb{R}^d$ mapping that maps a point $\mathbf{x}$ in the network area ϕ to a point $\mathbf{x}$ in a d-dimensional space $\mathbb{R}^d$. Consider z as a linear function of $\mathbf{y} = \Phi(\mathbf{x})$, *i.e.*,

$$z = w \cdot \mathbf{y} + b = w \cdot \Phi(\mathbf{x}) + b, \tag{5.2}$$

where $w \in \mathbb{R}^d$, $b \in \mathbb{R}$, and $\cdot$ denotes the inner product of two vectors. Then the canonical SVR formulation of this problem is [97]:

$$\begin{aligned} &\underset{w,b}{\text{minimize}} \quad \tfrac{1}{2}||w||^2, \\ &\text{subject to} \quad \begin{cases} z_k - (w \cdot \Phi(L_k) + b) < \epsilon_k \\ (w \cdot \Phi(L_k) + b) - z_k < \epsilon_k \ . \end{cases} \end{aligned} \tag{5.3}$$

The idea of this formulation is to regress the field intensity as a linear function of node locations in the new space $\mathbb{R}^d$.

This problem can be worked out by solving its dual problem with a quadratic programming formulation. It turns out that only the inner product $K(\mathbf{x}_1, \mathbf{x}_2) = \Phi(\mathbf{x}_1) \cdot \Phi(\mathbf{x}_2)$ in $\mathbb{R}^d$, but not $\Phi(\mathbf{x})$, is involved in the optimization procedure and in its result [97]. This greatly simplifies the optimization process as we do not need to know what the mapping actually is. $K(\mathbf{x}_1, \mathbf{x}_2)$ is called a *kernel*. With such a kernel SVR approach, $\widetilde{f}_i(\mathbf{x})$ can be represented by a set of locations $\{L_m\}_{\forall m \in \mathcal{SV}_i}$ (called *support*

vectors) and their weights $\{\alpha_m\}_{\forall m \in \mathcal{SV}_i}$, where $\mathcal{SV}_i$ is a subset of $\mathcal{W}_i$:

$$\widetilde{f}_i(\mathbf{x}) = \sum_{m \in \mathcal{SV}_i} z_m \alpha_m K(L_m, \mathbf{x}) - b. \tag{5.4}$$

In general, kernel SVR is to find out a subset of node locations (*i.e.*, support vectors) and node readings, together with the proper weights, which best represent the field intensity function with a kernel function.

Why using kernel SVR

Although SVR is proven effective in many disciplines (*e.g.*, [79, 85]), OACS is the first work that adopts kernel SVR for contour mapping in WSNs. In this section, we comprehensively justify the benefits to employ kernel SVR in providing a contour service.

Let us first illustrate the superiority of kernel SVR comparing with the scheme adopted in CME [118], a recent approach for WSN contour mapping which also suggests using similar machine learning approaches. CME formulates the problem of constructing a segment of contour line with value *cv* as a classification problem. The nodes with the readings larger than *cv* are in class A, while the nodes with the readings less than *cv* are in class B. Then a classification boundary can be calcu-

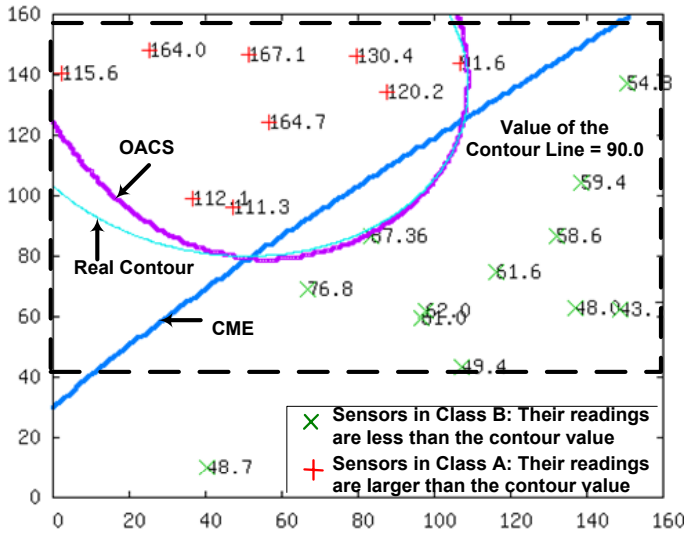


Figure 5.2: An example of the results generated by CME and OACS

lated by machine learning techniques and thus deemed as the contour line segment. Although such an approach is similar to ours since non-linear machine learning techniques are adopted, it relies simply on the shape of the distribution of the nodes' locations, while disregarding the sensor readings in the same class. As a result, it cannot generate satisfactory contour lines.

Figure 5.2 shows a typical result generated by CME. First, CME generates the same curve for the requests with values between 80.0 and 90.0. This is because in this example scenario class A and class B remain the same if the values of the requested contour line are between 80.0 and 90.0, and consequently, the classification boundaries (*i.e.*, the contour line) calculated by CME are the same. Second, CME tends to generate a biased

contour line. This is due to the even distribution of the nodes in class A and class B in the top-left part and the bottom-right part of the dotted rectangle, respectively. CME does not consider the values of sensor readings. As a result, the nodes in class B in the top-right and bottom-left corners, although with smaller readings, attract the classification boundary to go along top-right to bottom-left. OACS with kernel SVR which takes into account the sensor readings but not merely their locations, on the contrary, does not face these algorithmic inadequacies. Hence it can generate a more accurate result as demonstrated in the figure.

Besides its advantages comparing with the scheme adopted in CME, kernel SVR has many merits *per se*. First of all, it can equip OACS with a flexible method to deal with L-enquiries and M-enquiries. Examining its theoretical foundation, we can find that kernel SVR is a scheme that learns an output function based merely on its input samples, *i.e.*, $\mathcal{W}_i$. Therefore, by controlling the set of input samples, we can adapt it well to L-enquiries or M-enquiries. For L-enquiries, since we require $\widetilde{f}_i(x)$ to approximate $f(x)$ well in *only* the area around the contour line, $\mathcal{W}_i$ can be selected so that $\{z_k\}_{\forall k \in \mathcal{W}_i}$ are close to the contour value. Whereas for M-enquiries, many contour lines are required simul-

taneously and they are spreading throughout the cluster area; therfore, the regression result is important anywhere of the network. Hence a larger $\mathcal{W}_i$ with the nodes more evenly distributed can be the input for kernel SVR. Adaptively serving both types of enquires is a novel consideration of OACS in contrast to the current approaches which aim at either drawing a contour line efficiently or drawing a contour map efficiently, but not both.

Second, kernel SVR can conveniently handle the non-linear nature of contour lines. Observe from Equation (5.4) that $\widetilde{f}_i(\mathbf{x})$ is a combination of kernel functions $K(L_m, \mathbf{x})$. If we want to represent the field as a combination of polynomial functions with degree d, we only need to assign a polynomial kernel $K(\mathbf{x}_1, \mathbf{x}_2)$= $(\mathbf{x}_1 \cdot \mathbf{x}_2 + 1)^d$ for the kernel-SVR. Similarly, if we want to represent the field as a combination of Gaussian functions, we just need a Radial Basis Function (RBF) kernel with the form $K(\mathbf{x}_1, \mathbf{x}_2)$=$e^{-\gamma||\mathbf{x}_1-\mathbf{x}_2||^2}$, where e is the base of the natural logarithm. This provides a user the flexibility to select different non-linear curve functions according to the features of different sensing applications. Most importantly, the non-linearity of the results can thus be easily generated by applying the kernel method without causing much computational burden. We expect that the computational load of SVR is acceptable for a

cluster head equipped with the existing sensor platforms such as Crossbow Imote2 with a 400MHz MCU [19], since it just needs to solve a small-scale quadratic programming problem.

Finally, the result of kernel SVR is simple in representation. According to Equation 5.4, $\widetilde{f}_i(\mathbf{x})$ can be represented by a set of support vectors and their weights. A packet of a few tens of bytes is enough to encapsulate the result. OACS favors such a simple result since a cluster head hence only needs to send a single packet to the sink, which avoids high communication costs.

5.4.2 Enhancing Precision via Active Node Selection Schemes

Since there is no *a priori* knowledge on the requested contour line/map, initially, OACS lets λ percentage of nodes be on duty and then calculates the contour line/map based on these nodes. If the precision of the result does not match the user requirement (*i.e.*, λ<p), it will open a set of sleeping nodes and re-calculate the results based on the newly-added readings from this set of nodes.

Let $\mathcal{D}_i$ denote the set of the sleeping nodes in cluster i, *i.e.*, $\mathcal{S}_i$=$\mathcal{D}_i \cup \mathcal{W}_i$. A straightforward way to select the set of to-be-

opened nodes is to *randomly* select $(\lambda - p)n_i$ nodes from $\mathcal{D}_i$ and recalculate the contour line/map based on their readings. However, this scheme is inefficient. For example, for an L-enquiry, it may select the nodes with readings far away from the contour line value requested, but these nodes are almost useless. For an M-enquiry, it may select two nodes close to each other, but they are redundant. To avoid such situations, the way to select these $(\lambda - p)n_i$ nodes should be carefully studied.

For classification problems in machine learning, a similar concern is how to select a set of training samples for enhancing the classification precision. Many active learning schemes have been proposed in recent approaches which actively select training samples for text [50, 106] or image classification problems [15]. Although these algorithms are specifically designed for classification problems, their underlying idea, *i.e.*, selecting the samples that can provide most information, sheds light on solving our problem. We hence tailor two active learning schemes for processing L-enquiries and M-enquiries to select $(\lambda - p)n_i$ nodes, as described in what follows.

L-enquiry case

Consider the space $\mathbb{R}^d$ where the locations of nodes are projected by $\Phi : \mathbb{R}^2 \mapsto \mathbb{R}^d$. The contour line with value cv found by kernel SVR is hence an estimated hyperplane P_e in $\mathbb{R}^d$, where the points inside satisfy:

$$w \cdot \mathbf{y} + b = cv \Rightarrow w \cdot \Phi(\mathbf{x}) + b - cv = 0. \tag{5.5}$$

Since P_e is an estimation of the actual hyperplane P_a mapped from the actual contour line, it inevitably deviates from P_a. Let us consider a point $\mathbf{y}_1$ that is to one side of P_e and let d denote the distance from $\mathbf{y}_1$ to P_e. Note that the closer $\mathbf{y}_1$ is to P_e, the more likely $\mathbf{y}_1$ may actually be to the other side of P_a, *i.e*, the more uncertain for us to know whether $\mathbf{y}_1$ is actually on which side of P_a.

The distance d_j in $\mathbb{R}^d$ between a node s_j to the estimated hyperplane P_a is:

$$d_j = \left| \frac{w \cdot \Phi(L_j) + b - cv}{w} \right| \tag{5.6}$$

We then select a node $s_t \in \mathcal{D}_i$ as a to-be-opened node if it is the closest sleeping node to the hyperplane since the uncertainty of whether the node is to the other side of the actual contour

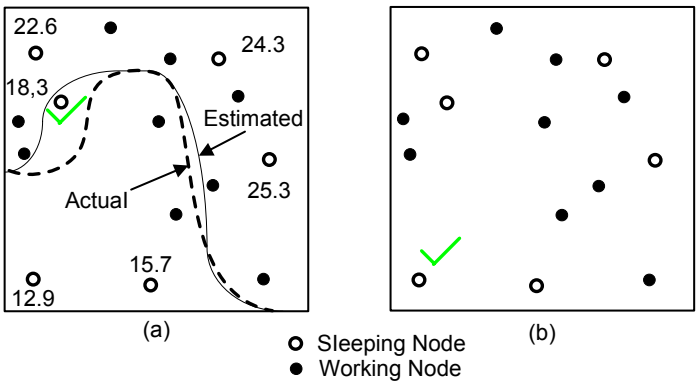
line is the maximum. In other words, its reading can potentially change the shape of the calculated contour line the most, and thus can best improve the accuracy of kernel SVR at the locations around the actual contour line. Formally,

$$\begin{aligned} t = & \operatorname*{argmin}_{\forall j \in \mathcal{D}_i} d_j = \operatorname*{argmin}_{\forall j \in \mathcal{D}_i} \left| \frac{w \cdot \Phi(L_j) + b - cv}{w} \right| \\ \Rightarrow & \quad t = \operatorname*{argmin}_{\forall j \in \mathcal{D}_i} |w \cdot \Phi(L_j) + b| \\ \Rightarrow & \quad t = \operatorname*{argmin}_{\forall j \in \mathcal{D}_i} \left| \widetilde{f}_i(L_j) \right| \end{aligned} \tag{5.7}$$

where the first transform is because $|w|$ and cv are the same for all nodes in $\mathcal{D}_i$ and the second is based on the mechanism of kernel SVR. See Figure 5.3(a) for an example where the value of the contour line requested is 20.0. The sleeping node, where the *estimated* field intensity 18.3 is closest to the contour value 20.0, should be opened to obtain its reading. This is because it is the most uncertain one regarding whether it is to which side of the actual contour line.

Note that only $\widetilde{f}_i(\mathbf{x})$ is involved in solving Equation (5.7). This is very convenient since the $\widetilde{f}_i(\mathbf{x})$ in Equation (5.4) estimated by kernel SVR is already at hand based on the readings of $\mathcal{W}_i$.

Then s_t is turned on and added to $\mathcal{W}_i$. Kernel SVR runs



(a) In processing an L-enquiry, the sleeping node, where the *estimated* field intensity 18.3 is the closest to the contour value 20.0, should be selected as a to-be-opened node. This is because it is the most uncertain regarding whether the node is to which side of the actual contour line. (b) In processing an M-enquiry, the sleeping node in the bottom-left corner is the farthest from the on-duty nodes. It is the most uncertain one for the field intensity at that location. Hence it should be opened to obtain its reading.

Figure 5.3: Demonstration of the active node selection schemes

again to update the contour line. The above procedure can be iterated until the precision of the generated contour line reaches the requirement, *i.e.*, pn_i nodes have been considered in producing the contour line.

M-enquiry case

For processing an M-enquiry, the situation is a little bit different from how to process an L-enquiry discussed in Section 5.4.2. What we care now is to enhance the precision of kernel SVR everywhere in the cluster area. We discuss how to select a to-be-opened node as follows.

Consider again the space $\mathbb{R}^d$. The distance d_{kj} between two

nodes s_k and s_j in $\mathbb{R}^d$ is:

$$\begin{aligned} d_{kj} &= ||(\Phi(L_k) - \Phi(L_j)||^2 \\ &= \sqrt{K(L_k, L_k) + K(L_j, L_j) - 2K(L_k, L_j)} \end{aligned} \tag{5.8}$$

Since $\widetilde{f}_i(\mathbf{x})$ is predicted by kernel SVR with the readings of the on-duty nodes, the farther a location $\mathbf{x}$ from the on-duty nodes is, the more uncertain the $\widetilde{f}_i(\mathbf{x})$ is. Therefore, to improve the accuracy of $\widetilde{f}_i(\mathbf{x})$, we should choose to open a sleeping node far away from the on-duty nodes. We define the distance between a sleeping node to $\mathcal{W}_i$ as the minimum distance between it and all nodes in $\mathcal{W}_i$, *i.e.*, in $\mathbb{R}^d$ it is $\min_{\forall k \in \mathcal{W}_i} d_{kj}$. We select a sleeping node $s_t \in \mathcal{D}_i$ that has the maximum distance to $\mathcal{W}_i$. Formally,

$$t = \underset{\forall j \in \mathcal{D}_i}{\operatorname{argmax}}(\min_{\forall k \in \mathcal{W}_i} d_{kj}) \tag{5.9}$$

Note that if we use an RBF kernel in SVR, $K(\mathbf{x}, \mathbf{x}) = e^{-\gamma||\mathbf{x}-\mathbf{x}||^2} = 1$. Equation (5.9) can be reduced to:

$$\begin{aligned} t &= \underset{\forall j \in \mathcal{D}_i}{\operatorname{argmax}}(\min_{\forall k \in \mathcal{W}_i} (-2K(L_k, L_j))) \\ &= \underset{\forall j \in \mathcal{D}_i}{\operatorname{argmax}}(\min_{\forall k \in \mathcal{W}_i} ||L_k - L_j||), \end{aligned} \tag{5.10}$$

which is exactly the same as selecting a sleeping node that has the maximum distance to $\mathcal{W}_i$ in the original space $\mathbb{R}^2$. See Figure 5.3(b) for an example where the sleeping node in the bottom-left corner is the farthest from the on-duty nodes. It is the most uncertain one for the field intensity at that location. Hence it should be opened to obtain its reading.

5.5 Performance Study

To study the effectiveness of OACS in addressing the WSN contour mapping problem, we simulate a WSN. Without loss of generality, clusters are formed by evenly dividing the network into grids and their heads are randomly selected, which is similar to the scheme in [118]. The grid numbers are also selected based on the scheme in [118]. A widely-adopted SVR solver in SVM$^{\text{light}}$ [54] is employed to solve our kernel SVR problems for cluster heads[11].

The field intensity is generated by the model adopted in [75, 118], which is briefed as follows. M sources are randomly distributed in the network area, the locations of which are denoted by $\mathcal{T}=\{T_m\}_{m=1}^{M}$. Then the field intensity $f(\mathbf{x})$ at location

[11]The solver is implemented in C. We expect it convenient to port the solver to some existing operating systems of sensor platforms, *e.g.* LiteOS [11]. But we have not studied this issue in this thesis. It needs extensive future work, though.

$\mathbf{x}$ is determined by the summation of the diffusion from the sources:

$$f(\mathbf{x}) = \sum_{T_m \in \mathcal{T}} \frac{1}{(k||\mathbf{x} - T_m|| + 1)^{\alpha}}. \tag{5.11}$$

Note that this is a realistic model since in general the field is a cumulative result by several sources (*e.g.*, those that emit heat) while the effect of a source decays exponentially with a factor larger than 2 in space. The reading of a sensor s_i is then given by $f(L_i) + \epsilon$.

Table 5.1: Simulation settings in Chapter 5

Area of sensor field	320m × 320m
Rode deployment scheme	Randomly deployed in a uniform manner
Communication range R_c	30m
Decay factor α and k	3 and 0.01
Packet size	48 bytes

The details of our network settings are shown in Table 5.1, which are typical WSN settings similar to those adopted in [118]. Accuracy of a contour line/map is calculated based on the MSE of the results with respect to the actual line/map. In our following performance study, for each setting we adopt different random seeds in every run and the results are averaged.

5.5.1 Advantages of Kernel SVR

We first study the accuracy of our OACS in processing the L-enquiries, comparing with the CME approach proposed in [118]. We adopt CME for comparison purpose because it is the most up-to-date contour mapping approach, which also employs the state-of-the-art machine learning techniques to handle the non-linear nature of contour line.

For each L-enquiry $[cv, p]$, cv is a randomly-selected field intensity value. We set p=100% for fairness consideration as in this case OACS would not take any advantages of its active node-selection schemes. Our purpose is to compare the performance of our kernel-SVR algorithm and the classification algorithm adopted in CME.

We change the node density in processing L-enquiries. Figure 5.4 demonstrates an example result where M=3. We can see that OACS greatly outperforms CME especially when the node density is low. This is not surprising: CME relies only on the node-location distribution. It can generate accurate results only if the node density is very high. In contrast, OACS considers sensor readings in its kernel-SVR algorithm, which adapts well to low node density.

Note that the energy required for processing an L-enquiry

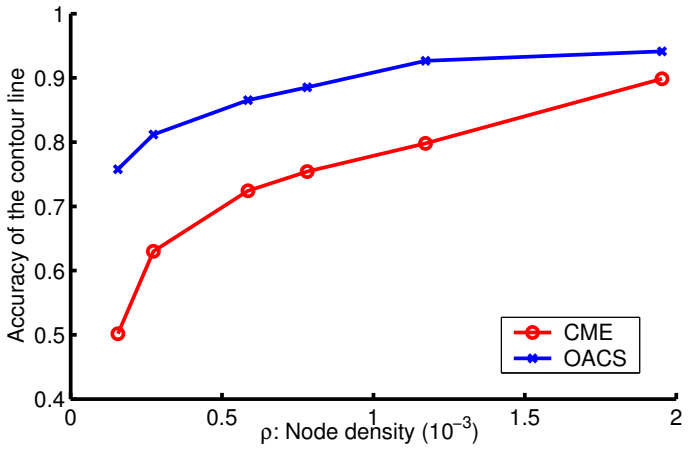


Figure 5.4: Accuracy comparisons between OACS and CME in processing L-enquiries

for both CME and OACS is almost the same in this experiment (where the precision requirement p=100%). This is because both approaches let the nodes with readings close to the contour value report to their cluster heads.

We then compare OACS and CME in handling M-enquiries. Similar to the previous experiment, we also set p=100%. We change the node density and study the accuracy of the maps generated by OACS and CME. The accuracy results are demonstrated in Figure 5.5 where we randomly place six sources (*i.e.*, M=6) in the network. It shows that OACS also outperforms CME. The total number of packets that need to be sent by all in-network nodes are compared in Figure 5.6. We can see that if we need to generate a contour map with more contour lines, CME spends more energy than OACS does since CME has to

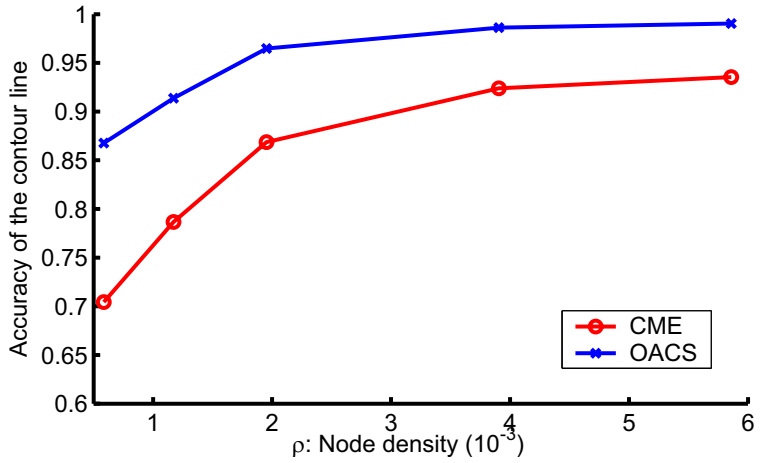


Figure 5.5: Accuracy comparisons between OACS and CME in processing M-enquiries

calculate each contour line separately. In contrast, the number of packets remains constant for OACS since it generates the same result for an M-enquiry with the same p.

These simulations show that OACS outperforms CME in processing both L-enquiries and M-enquires in terms of accuracy of the results, while OACS is more energy-efficient when calculating the contour map. Note that in these simulations, we set p=100%. In case that the user does not need such a high p, obviously OACS can conserve much more energy since it has a scheme to let some nodes sleep. CME, however, lacks such a scheme and as a result all nodes have to be on duty. Hence, when p<100%, the energy consumption results are omitted since the better performance of OACS is straightforward. In conclusion, we verify the advantages of employing kernel SVR in contour

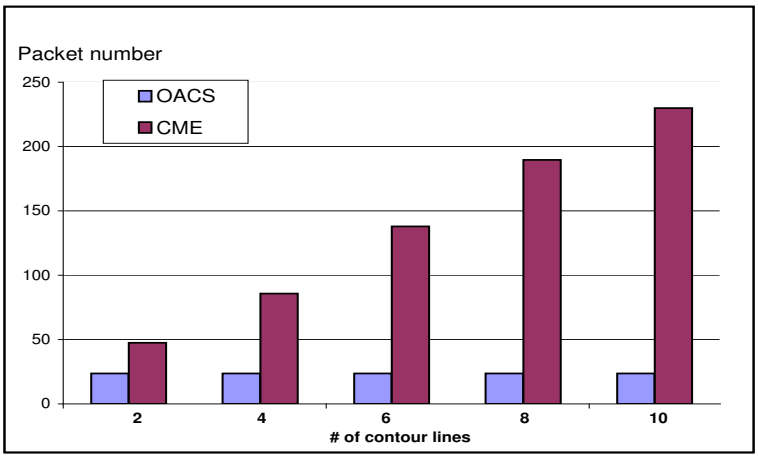


Figure 5.6: Total packet numbers sent by OACS and CME in processing M-enquiries

mapping.

5.5.2 Advantages of Active Node Selection Scheme

Now we investigate how well our active node selection scheme performs in handling different user requirements on the precisions of the contour line/map requested. We randomly place six sources (*i.e.*, M=6) and deploy 400 nodes in the network area. Two approaches are adopted. One (denoted by OACS-Active) is OACS with our active node selection scheme. The other (denoted by OACS-Random) is another OACS version without such a scheme, where $(p\text{-}\lambda)n_i$ nodes are selected *randomly* from all the sleeping nodes. We set λ=12% for OACS, *i.e.*, initially, 12% of in-network nodes are on duty.

We study the accuracy of the results on L-enquiries and M-

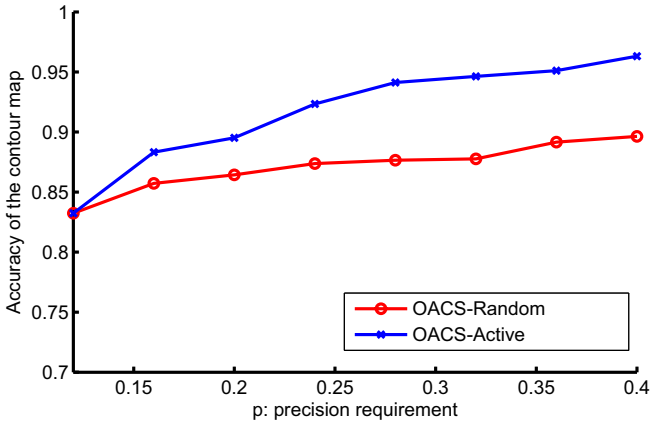


Figure 5.7: Accuracy as a function of precision requirement: L-enquiry case

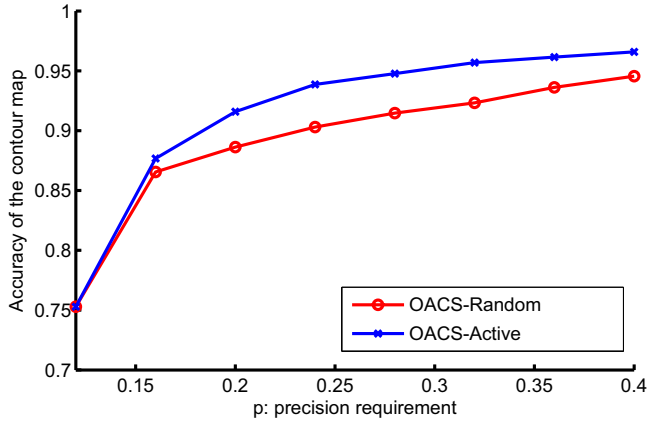


Figure 5.8: Accuracy as a function of precision requirement: M-enquiry case

enquiries, in which we change the precision requirement p from 12% to 40%. Figures 5.7 and 5.8 show the results. We can see that OACS with an active node scheme performs much better than that without such a scheme[12]. This is also a natural result since our active node selection scheme aims at minimizing the uncertainty of the generated contour lines/maps. Hence, it can

[12]Note that the accuracy results of both schemes when p=12% are the same as in this case p is equal to λ, which means that no sleeping nodes need to be turned on. As a result, the node selection schemes do not have any effects.

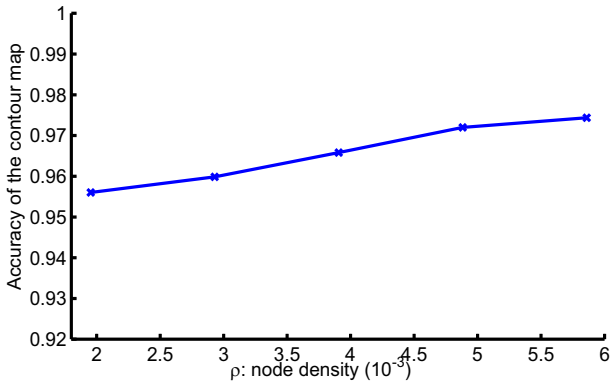


Figure 5.9: Accuracy as a function of node density

select a good set of sleeping nodes and open them for improving the results, which can greatly enhance the accuracy comparing with the random node selection scheme. We can instantly see that for achieving the same accuracy, OACS with an active node scheme can request to open less nodes than that without such a scheme. Consequently, OACS with an active node scheme can save more energy.

With the same setting, we further vary the node density and study how OACS with an active node selection scheme performs in processing M-enquiries. Figure 5.9 demonstrates the results, which show that the higher the node density, the more accurate the results. This is because when the node density is high, more candidates are available, and consequently a better set of to-be-opened nodes can be selected. Hence, the accuracy increases as the node density increases.

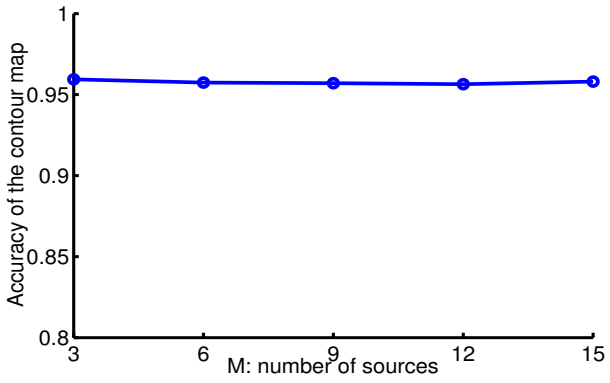


Figure 5.10: Accuracy as a function of source numbers

Finally, to study how OACS performs when the field intensity becomes more irregularly-distributed, we deploy 400 nodes in the network area and increase the number of sources M. Let OACS perform M-enquires with p=32%. Figure 5.10 shows that the accuracy of OACS almost remains unchanged when the source number increases. This demonstrates that the performance of OACS does not vary much with how the field intensity is distributed, which verifies the nice adaptivity attribute of OACS.

5.6 Conclusion

In this chapter, we presented OACS as a promising contour mapping tool in handling diverse user requests. We showed that OACS can handle the requests for both contour line and contour map energy-efficiently. Most importantly, it is intelligent

in energy saving when providing users the flexibility to request contour lines/maps with different precisions. Our work demonstrated that the active learning and the kernel SVR techniques from the machine learning field can be powerful tools for the contour mapping service in WSNs.

□ **End of chapter.**

Chapter 6

Conclusion

In-situ sensing with WSNs is a promising approach in environmental data collection and event detection. In this thesis, we investigated a variety of key problems in realizing WSN in-situ sensing systems. Since sensor nodes are powered by small batteries which are usually not rechargeable, our aim is to achieve energy-efficient WSNs given the low computation capacity of the sensor nodes.

For event detection WSNs, the coverage of the network, as a measure of event detection capability, should be maintained in an energy-efficient manner. We studied two coverage-oriented network scheduling problems in this thesis.

First, we addressed a critical coverage-oriented partitioning problem: how to divide a set of sensor nodes into a maximum number of disjoint subsets, so that each subset can cover the

entire network territory. We showed the interesting features of a fan-out index ι for a set of points. The application of ι to the coverage-oriented network partition problem was elaborated by employing it in a MAXINE algorithm. MAXINE exhibited good performance and short convergence time in our extensive simulation studies with a wide range of network settings and two general sensor coverage models. This provides a strong evidence that maximizing ι is a promising idea to reduce node redundancy. But there is still much room to further extend this research. For example, we are interested in introducing the connectivity requirements into the problem formulation so that the resulting subsets can guarantee certain network connectivity properties. Moreover, the distributed and localized version of MAXINE is actually conducted sequentially by one node after another. We are interested in have the algorithm executed by multiple nodes simultaneously, rather than in such a sequential manner.

Second, seamless system migration without downtime is necessary for WSNs that perform critical event detection tasks. Unfortunately, to our knowledge, this important problem has not been addressed in the literature. We presented the first formal study on this problem. We demonstrated that the downtime

can be eliminated by partitioning the sensors into a collection of subsets, and let each subset conduct the system migration tasks successively with the rest still performing normal event detection services. We proved the optimum partitioning of sensors in this context is NP-hard and then proposed a series of heuristics. We further extended our solution to a distributed implementation called the Sensor Network Reconfiguration Protocol (SNRP). Simulation results showed that these algorithms work well in various performance evaluations. Yet, we believe there is still room to extend this research. In particular, if the node locations are not available, we need to find a way to divide the sensors according to the in-network nodes' neighborhood information.

For data collection WSNs, we proposed a novel location-directed data transport protocols, namely, GDRP for conveying sensor-to-sink packets. GDRP aims to improve the energy efficiency of location-directed forwarding, so as to enhance the survivability of the network in practical deployment environments where network holes and barriers generally exist. We proved the performance guarantee of GDRP and verified its effectiveness in tolerating network holes and barriers with extensive simulation studies.

Furthermore, we presented OACS as a promising contour mapping tool in handling diverse user requests in data collection WSNs. We showed that OACS can handle the requests for both contour line and contour map energy-efficiently. Most importantly, it is intelligent in energy saving when providing users the flexibility to request contour lines/maps with different precision requirements. Our work demonstrated that the active learning and the kernel SVR techniques from the machine learning field can be powerful tools for the contour mapping service in WSNs. There are, however, many remaining issues to be explored. We are particularly interested in developing online algorithms for contour mapping, where a contour line/map can be incrementally polished based on new sensor readings instead of re-running kernel SVR. We also intend to investigate how OACS performs if we adopt a batch mode active learning scheme where all the to-be-opened nodes are selected simultaneously, instead of selecting the to-be-opened node one by one. Finally, how to calculate time-variant contour line/map is also of interest: A powerful contour service that can well exploit the temporal-correlations of sensor readings is still at large.

In conclusion, this thesis studies a bunch of protocols and algorithms to approach energy-efficient WSNs. Another critical

aspect of realizing WSN in-situ sensing is the implementation of reliable protocols and algorithms. WSNs software is quite different to traditional one. As a result, traditional software testing, debugging, and fault-tolerance techniques are hence inapplicable for WSN software, posing a great challenge for building reliable WSN in-situ sensing systems. In future work, we are also particularly interested in these issues in WSNs.

□ **End of chapter.**

Bibliography

[1] N. Ahmed, S. S. Kanhere, and S. Jha. The holes problem in wireless sensor networks: a survey. *ACM Mobile Computing and Communications Review*, 9(2):4–18, April 2005.

[2] A. Ailamaki, C. Faloutos, P. S. Fischbeck, M. J. Small, and J. VanBriesen. An environmental sensor network to determine drinking water quality and security. *Sigmod Record*, 32(4):47–52, 2003.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, April 2002.

[4] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the

sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605–634, 2004.

[5] G. Asada, T. Dong, F. Lin, G. Pottie, W. Kaiser, and H. Marcy. Wireless integrated network sensors: Low power systems on a chip. In *Proc. of the European Solid State Circuits Conference (ESSCIRC)*, The Hague, Netherlands, October 1998.

[6] F. Aurenhammer. Voronoi diagram: A survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(2):345–405, September 1991.

[7] X. Bai, S. Kumar, Z. Yun, D. Xuan, and T.-H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, Florence, Italy, May 2006.

[8] L. Blazevic, J.-Y. L. Boudec, and S. Giordano. A location-based routing method for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 4(2):97–110, March-April 2005.

[9] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Kluwer Wireless Networks*, 7(6):609–616, 2001.

[10] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low-cost outdoor localization for very small devices. *IEEE Personal Communication*, October 2000.

[11] Q. Cao, T. Abdelzaher, J. Stankovic, and T. He. The LiteOS operating system: Towards unix-like abstractions for wireless sensor networks. In *Proc. of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2008.

[12] Q. Cao and J. Stankovic. An in-field-maintenance framework for wireless sensor networks. In *Proc. of the IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2008.

[13] M. Cardei and D. Du. Improving wireless sensor network lifetime through power aware organization. *ACM Journal of Wireless Networks*, 11(3):333–340, May 2005.

[14] M. Cardei, D. MacCallum, X. Cheng, M. Min, X. Jia, D. Li, and D.-Z. Du. Wireless sensor networks with en-

ergy efficient organization. *Journal of Interconnection Networks*, 3(3-4):213–229, December 2002.

[15] E. Chang, S. Tong, K. Goh, and C.-W. Chang. Support vector machine concept-dependent active learning for image retrieval. *IEEE Transactions on Multimedia*, 2, 2005.

[16] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. SPAN: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Kluwer Wireless Networks*, 8(5):481–494, September 2002.

[17] J. Chen, S. Kher, and A. K. Somani. Distributed fault detection of wireless sensor networks. In *Proc. of the ACM Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks*, September 2006.

[18] W.-P. Chen, J. C. Hou, and L. Sha. Dynamic clustering for acoustic target tracking in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 3(3), 2004.

[19] Crossbow Technology. Imote2: High-performance wireless sensor network node. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf`.

[20] Crossbow Technology. Iris datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/IRIS_Datasheet.pdf`.

[21] D. Culler, D. Estrin, and M. Srivastava. Overview of sensor networks. *IEEE Computer Magazine*, 37(8):41–49, August 2004.

[22] J. C. Davis. Contouring algorithms. In *Proc. of the International Symposium on Computer-Assisted Cartography*, pages 352–359, Reston, VA, December 1975.

[23] K. A. Delin, R. Harvey, N. A. Chabot, S. Jackson, M. Adams, D. Johnson, and J. Britton. Sensor web in Antarctica: Developing an intelligent, autonomous platform for locating biological flourishes in cryogenic environments. In *Proc. of the Lunar and Planetary Science Conference*, Houston, TX, March 2003.

[24] K. A. Delin and S. P. Jackson. Sensor web for in situ exploration of gaseous biosignatures. In *Proc. of the IEEE Aerospace Conference*, volume 7, pages 465–472, March 2000.

[25] T. Dinh, W. Hu, P. Sikka, P. Corke, L. Overs, and S. Brosnan. Design and deployment of a remote robust sensor

network: Experiences from an outdoor water quality monitoring network. In *Proc. of the IEEE Confenrence on Local Computer Networks (LCN)*, pages 799–806, 2007.

[26] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, pages 155–161, 1996.

[27] M. F. Duarte and Y. H. Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, July 2004.

[28] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In *Proc. of the ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.

[29] G. Edmondo. Digital geologic field mapping using arcpad. In *Proc. of the Digital Mapping Techniques Workshop, U.S. Geological Survey Open-File Report 02-370*, 2002.

[30] D. Estrin, D. Culler, and K. Pister. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1(1), January-March 2002.

[31] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2001.

[32] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proc. of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Seattle, Washington, August 1999.

[33] Exscal Research Group, Ohio State University. *ExScal: Extreme Scale Wireless Sensor Networking.* `http://cast.cse.ohio-state.edu/exscal/`.

[34] Q. Fang, J. Gao, and L. J. Guibas. Locating and bypassing holes in sensor networks. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*, 2004.

[35] E. Felemban, C.-G. Lee, E. Ekici, R. Boder, and S. Vural. Probabilistic QoS guarantee in reliability and timeliness domains in wireless sensor networks. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*, Miami, FL, U.S., March 2005.

[36] R. Flury, S. V. Pemmaraju, and R. Wattenhofer. Greedy routing with bounded stretch. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*, 2009.

[37] Free Software Foundation. The GLPK (GNU Linear Programming Kit). `http://www.gnu.org/software/glpk/`.

[38] H. Frey and I. Stojmenovic. On delivery guarantees of face and combined greedy-face routing algorithms in ad hoc and sensor networks. In *Proc. of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pages 390–401, Los Angeles, CA, September 2006.

[39] P. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning tree. *ACM Transactions on Programming Languages and Systems*, 5(5):66–77, January 1983.

[40] S. Gandhi, J. Hershberger, M. Graphics, and S. Suri. Approximate isocontours and spatial summaries for sensor networks. In *Proc. of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2007.

[41] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanner for routing in mobile networks. In *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, Long Beach, CA, USA, October 2001.

[42] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., New York City, NY, 1979.

[43] L. Gu and J. A. Stankovic. Radio-triggered wake-up for wireless sensor networks. *Kluwer Journal of Real-Time Systems*, 29(2-3):157–182, March 2005.

[44] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden. Distributed regression: an efficient framework for modeling sensor network data. In *Proc. of the ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004.

[45] J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2:84–90, 1960.

[46] J. M. Hammersley. Monte-Carlo methods for solving multivariable problems. *Annals of the New York Academy of Science*, 86:844–874, 1960.

[47] T. He, J. Stankovic, C. Lu, and T. Abdelzaher. SPEED: A realtime routing protocol for sensor networks. In *Proc. of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, page 46C55, Providence, RI, U.S., May 2003.

[48] J. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Towards sophisticated sensing with queries. In *Proc. of the ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN)*, Washington D.C., November 2005.

[49] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proc. of the International Conference on Architectural Support for Programming Languages and Operating Systems*, Cambridge, MA, November 2000.

[50] S. C.-H. Hoi, R. Jin, and M. R. Lyu. Large-scale text categorization by batch mode active learning. In *Proc. of*

the International World Wide Web Conference (WWW), Edinburgh, England, UK, 2006.

[51] H. Huang. Adaptive algorithms to mitigate inefficiency in greedy geographical routing. *IEEE Communication Letters*, 10(3):150–152, March 2006.

[52] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Boston, Massachusetts, August 2000.

[53] R. Jain, W. Hawe, and D. Chiu. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *Technical Report DEC-TR-301*, September 1984.

[54] T. Joachims. Making large-scale svm learning practical. In *Advances in Kernel Methods - Support Vector Learning (eds. B. Schölkopf, C. Burges and A. Smola)*, MIT-Press, 1999.

[55] E. Karnakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proc. of the Canadian Confer-*

ence on Computational Gemetry (CCCG), pages 51–54, Vancouver, Canada, Aug 1999.

[56] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proc. of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pages 243–254, Boston, MA, 2000.

[57] R. Kershner. The number of circles covering a set. *American Journal of Mathematics*, 61:665–671, 1939.

[58] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Proc. of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 254–263, 2007.

[59] D. Krige. *A statistical approach to some mine valuations and allied problems at the Witwatersrand*. Master's thesis of the University of Witwatersrand, 1951.

[60] L. Krishnamurthy. Wireless sensor networks in intel fabrication plants. `http://www.intel.com/research/vert_manuf_condmaint.htm`, May 2004.

[61] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis. Design and deployment of industrial sensor networks: experiences from a semiconductor plant in the north sea. In *Proc. of the ACM International Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 64–75, 2005.

[62] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad hoc routing: of theory and practice. In *Proc. of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2003.

[63] B. Leong, B. Liskov, and R. Morris. Geographic routing without planarization. In *Proc. of the USENIX Symposium on Networked Systems Design & Implementation (NSDI'06)*, 2006.

[64] P. Levis. Tinyos programming manual. `http://www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf`, 2008.

[65] X.-Y. Li, Y. Wang, P.-J. Wan, W.-Z. Song, and O. Frieder. Localized low-weight graph and its applications in wireless ad hoc networks. In *Proc. of the IEEE Conference*

on Computer Communications (INFOCOM), Hong Kong, March 2004.

[66] B. Liu and D. Towsley. A study on the coverage of large-scale sensor networks. In *Proc. of the IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, Fort Lauderdale, FL, October 2004.

[67] H. Liu, X. Jia, P. Wan, C.-W. Yi, S. K. Makki, and N. Pissinou. Maximizing lifetime of sensor surveillance systems. *IEEE/ACM Transactions on Networking*, 15(2):334–345, April 2007.

[68] Y. Liu and M. Li. Iso-map: Energy-efficient contour mapping in wireless sensor networks. In *Proc. of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 36–44, Toronto, Canada, June 2007.

[69] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He. RAP: A real-time communication architecture for largescale wireless sensor networks. In *Proc. of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, San Jose, CA, U.S., September 2002.

[70] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proc. of the ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[71] W. Manges, G. Allgood, and S. Smith. It's time for sensors to go wireless. part 1: Technological underpinnings. *Sensors Magazine*, April 1999.

[72] W. Manges, G. Allgood, and S. Smith. It's time for sensors to go wireless. part 2: Take a good technology and make it an economic success. *Sensors Magazine*, May 1999.

[73] P. J. Marron, A. Lachenmann, D. Minder, M. Gauger, O. Saukh, and K. Rothermel. Management and configuration issues for sensor networks. *International Journal on Network Management*, 15(4):235–253, July 2005.

[74] S. Megerian, F. Koushanfar, G. Qu, G. Veltri, and M. Potkonjak. Exposure in wireless sensor networks: Theory and pratical solutions. *ACM Journal of Wireless Networks*, 8(5):443–454, 2002.

[75] X. Meng, T. Nandagopal, L. Li, and S. Lu. Contour maps: Monitoring and diagnosis in sensor networks. *Computer Networks*, 50(15):2820–2838, 2006.

[76] R. Min, M. Bhardwaj, S. Cho, E. Shih, A. Sinha, A. Wang, and A. Chandrakasan. Low-power wireless sensor networks. In *Proc. of the International Conference on VLSI Design*, Bangalore, India, January 2001.

[77] E. Ngai, Y. Zhou, M. Lyu, and J. Liu. Reliable reporting of delay-sensitive events in wireless sensor-actuator networks. In *Proc. of the IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, Vancouver, Canada, October 2006.

[78] E. Ngai, Y. Zhou, M. Lyu, and J. Liu. LOFT: A latency-oriented fault tolerant transport protocol for wireless sensor-actuator networks. In *Proc. of the IEEE Global Communications Conference (GlobeCom)*, Washington, DC, USA, November 2007.

[79] K. S. Ni and T. Q. Nguyen. Image superresolution using support vector regression. *IEEE Transactions on Image Processing*, 16(6):1596–1610, June 2007.

[80] R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. In *Proc. of the International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 80–95, Palo Alto, CA, April 2003.

[81] P. Padhy, K. K. Martinez, A. Riddoch, H. Ong, and J. Hart. Glacial environment monitoring using sensor networks. In *Proc. of the Workshop on Real-World Wireless Sensor Networks (REALWSN)*, Stockholm, Sweden, June 2005.

[82] J. Paek, K. Chintalapudi, J. Cafferey, R. Govindan, and S. Masri. A wireless sensor network for structural health monitoring: Performance and experience. In *Proc. of the IEEE Workshop on Embedded Networked Sensors (EmNetS)*, Syndney, Australia, May 2005.

[83] G. Pottie and W. Kaiser. Wireless integrated network sensors. *Communications of ACM*, 43(5), May 2000.

[84] R. C. Prim. Shortest connection networks and some generalisations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[85] J. Qiu, W. Sheffler, D. Baker, and W. S. Noble. Ranking predicted protein structures with support vector regression. *Proteins: Stucture, Function, and Bioinformatics*, 71(3):1175–1182, 2007.

[86] R. Ramanathan and R. Rosales-Hain. Topology control of multihop wireless networks using transmit power adjust-

ment. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*, Tel Aviv, Israel, March 2002.

[87] K. Römer and F. Mattern. The design space of wireless sensor networks. *IEEE Wireless Communications*, 11(6):54–61, December 2004.

[88] V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 17(8):1333–1344, 1999.

[89] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. ESRT: event to sink reliable transport in wireless sensor networks. In *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, June 2003.

[90] P. Santi. The critical transmitting range for connectivity in mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 4(3), May-June 2005.

[91] P. Santi, D. Blough, and F. Vainstein. A probabilistic analysis for the range assignment problem in ad hoc networks. In *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 212–220, Long Beach, October 2001.

[92] L. Schwiebert, S. K. S. Gupta, and J. Weinmann. Research challenges in wireless networks of biomedical sensors. In *Proc. of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pages 151–165, 2001.

[93] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter. Luster: Wireless sensor network for environmental research. In *Proc. of the ACM International Conference on Embedded Networked Sensor Systems (SENSYS)*, Sydney, Australia, November 2007.

[94] C. Sharp, S. Shaffert, A. Woo, N. Sastry, C. Karlof, S. Sastry, and D. Culler. Design and implementation of a sensor network system for vehicle tracking and autonomous interception. In *Proc. of the European Conference on Wireless Sensor Networks (EWSN)*, 2005.

[95] A. Silberstein, R. Braynard, and J. Yang. Constraint chaining: On energyefficient continuous monitoring in sensor networks. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, Chicago, IL, 2006.

[96] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *Proc. of the IEEE International Conference on Communications (ICC)*, volume 2, Helsinki, Finland, June 2001.

[97] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[98] I. Solis and K. Obraczka. Efficient continuous mapping in sensor networks using isolines. In *Proc. of the Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, San Diego, CA, July 2005.

[99] S. Srinivasan, K. Ramamritham, and P. Kulkarni. ACE in the hole: Adaptive contour estimation using collaborating mobile sensors. In *Proc. of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2008.

[100] M. B. Srivastava, R. R. Muntz, and M. Potkonjak. Smart kindergarten: sensorbased wireless networks for smart developmental problem-solving enviroments. In *Proc. of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pages 132–138, 2001.

[101] F. Stann and J. Heidemann. RMST: reliable data transport in sensor networks. In *Proc. of the IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003.

[102] G. Tan, M. Bertier, and A.-M. Kermarrec. Convex partition of sensor networks and its use in virtual coordinate geographic routing. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*, 2009.

[103] D. Tian and N. D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *Proc. of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 32–41, Atlanta, Georgia, 2002.

[104] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman. Infrastructure tradeoffs for sensor networks. In *Proc. of the ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, GA, USA, September 2002.

[105] TinyOS Community Forum. Tinyos: An open-source os for the networked sensor regime. `http://www.tinyos.net`.

[106] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

[107] M.-J. Tsai, F.-R. Wang, H.-Y. Yang, and Y.-P. Cheng. Virtualface: An algorithm to guarantee packet delivery of virtual-coordinate-based routing protocols in wireless sensor networks. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*, 2009.

[108] C. Wan, A. Campbell, and L. Krishnamurthy. Psfq: A reliable transport protocol for wireless sensor networks. In *Proc. of the ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, September 2002.

[109] P.-J. Wan, C.-W. Yi, F. Yao, and X. Jia. Asymptotic critical transmission radius for greedy forwarding routing in wireless ad hoc networks. In *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, Florence, Italy, May 2006.

[110] Q. Wang, Y. Zhu, and L. Cheng. Reprogramming wireless sensor networks: Challenges and approaches. *IEEE Network*, pages 48–55, May-June 2006.

[111] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Proc. of the ACM International Conference on Embedded Networked Sensor Systems (SENSYS)*, Los Angeles, CA, November 2003.

[112] R. Wattenhofer, P. Bahl, L. Li, and Y. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*, April 2001.

[113] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proc. of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Seattle, USA, 2006.

[114] K. Whitehouse. The design of calamari: an ad-hoc localization system for sensor networks. M.Sc. Thesis, UC Berkeley, 2002.

[115] R. Williams. *The Geometrical Foundation of Natural Structure: A Source Book of Design.* Dover Publications Inc., pp. 51-52. New York, 1979.

[116] G. Xing, C. Lu, R. Pless, and J. A. O'Sullivan. Co-Grid: an efficient converage maintenance protocol for distributed sensor networks. In *Proc. of the ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN)*, Berkeley, CA, April 2004.

[117] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proc. of the ACM International Conference on Embedded Networked Sensor Systems (SENSYS)*, Baltimore, MD, November 2004.

[118] Y. Xu, W.-C. Lee, and G. Mitchell. CME: a contour mapping engine in wireless sensor networks. In *Proc. of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 133–140, 2008.

[119] W. Xue, Q. Luo, L. Chen, and Y. Liu. Contour map matching for event detection in sensor networks. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, Chicago, IL, 2006.

[120] T. Yan, T. He, and J. A. Stankovic. Differentiated surveillance for sensor networks. In *Proc. of the ACM Interna-*

tional Conference on Embedded Networked Sensor Systems (SENSYS), Los Angeles, CA, November 2003.

[121] Z. Yang, M. Li, and Y. Liu. Sea depth measurement with restricted floating sensors. In *Proc. of the IEEE Real-Time Systems Symposium (RTSS)*, 2007.

[122] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *Proc. of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, Providence, Rhode Island, May 2003.

[123] O. Younis and S. Fahmy. HEED: a hybrid, energy-efficient. distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3(4):336–379, 2004.

[124] G. Zhao, X. Liu, and M.-T. Sun. Anchor-based geographic routing for sensor networks using projection distance. In *Proc. of the IEEE International Symposium on Wireless Pervasive Computing (ISWPC)*, San Juan, Puerto Rico, Febrary 2007.

[125] J. Zhao, R. Govindan, and D. Estrin. Residual energy scans for monitoring wireless sensor networks. In *Proc. of*

the IEEE Wireless Communications and Networking Conference (WCNC), pages 17–22, March 2002.

[126] Y. Zhou. Energy-efficient reliable wireless sensor networks. M.Phil. Thesis, The Chinese University of Hong Kong, 2006.

[127] Y. Zhou, M. Lyu, and J. Liu. On setting up energy-efficient paths with transmitter power control in wireless sensor networks. In *Proc. of the IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, pages 440–448, Washington, DC, USA, November 2005.

[128] Y. Zhou, M. Lyu, and J. Liu. An index-based sensor-grouping mechanism for field coverage wireless sensor networks. In *Proc. of the IEEE International Conference on Communications (ICC)*, Beijing, China, May 2008.

[129] Y. Zhou, M. Lyu, and J. Liu. On sensor network reconfiguration problem for downtime-free system migrations. In *Proc. of the ICST International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine)*, Hong Kong, China, July 2008.

[130] Y. Zhou, M. Lyu, and J. Liu. On sensor network reconfiguration for downtime-free system migration. *ACM/Springer*

Mobile Networks and Applications, 14(2):241–252, April 2009.

[131] Y. Zhou, M. Lyu, and J. Liu. Surviving holes and barriers in geographic data reporting for wireless sensor networks. In *Proc. of the IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, Macau, China, October 2009.

[132] Y. Zhou, M. Lyu, J. Liu, and H. Wang. PORT: A price-oriented reliable transport protocol for wireless sensor networks. In *Proc. of the IEEE International Symposium on Software Reliability Engineering (ISSRE)*, pages 117–126, Chicago, IL, USA, November 2005.

[133] Y. Zhou and M. R. Lyu. An energy-efficient mechanism for self-monitoring sensor web. In *Proc. of the 28th IEEE Aerospace Conference*, Big Sky, MT, March 2007.

[134] Y. Zhou, E. Ngai, M. Lyu, and J. Liu. POWER-SPEED: A power-controlled real-time data transport protocol for wireless sensor-actuator networks. In *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*, Hong Kong, China, March 2007.

[135] Y. Zhou, J. Xiong, M. Lyu, J. Liu, and K.-W. Ng. Energy-efficient on-demand contour service for wireless sensor networks. In *Proc. of the IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, Macau, China, October 2009.

[136] Y. Zhou, H. Yang, M. Lyu, and E. Ngai. A point-distribution index and its application to sensor grouping in wireless sensor networks. In *Proc. of the International Wireless Communications and Mobile Computing Conference (IWCMC)*, Vancouver, Canada, July 2006.

[137] M. Zorzi and R. R. Rao. Geographic random forwarding (GeRaF) for ad hoc and sensor networks: Energy and latency performance. *IEEE Transactions on Mobile Computing*, 2(4):349–365, Oct.-Dec. 2003.

[138] M. Zorzi and R. R. Rao. Geographic random forwarding (GeRaF) for ad hoc and sensor networks: Multihop performance. *IEEE Transactions on Mobile Computing*, 2(4):337–347, Oct.-Dec. 2003.