Bang Wang

Coverage Control in Sensor Networks

Springer

Dr. Bang Wang
Nanyang Technological University
Intelligent Systems Centre
50 Nanyang Drive
Research Techno Plaza, Level 7
Singapore 637553
Singapore
wangbang@ntu.edu.sg

Series Editor
Professor A.J. Sammes, BSc, MPhil, PhD, FBCS, CEng
Centre for Forensic Computing
Cranfield University
DCMT, Shrivenham
Swindon SN6 8LA
UK

ISSN 1617-7975
ISBN 978-1-84996-058-8 e-ISBN 978-1-84996-059-5
DOI 10.1007/978-1-84996-059-5
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2010920018

Cover design: VTeX, Vilnius

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

To my wife Minghua, and our families

Preface

The advances in sensor design have decreased the size, weight, and cost of sensors by orders of magnitude, yet with the increase of higher spatial and temporal resolution and accuracy. With the fast progress of sensors design and communications technique, sensor networks have also been quickly evolving in both research and practical domains in the last decade. More and more sensor networks have been deployed in real-world to gather information for our daily life. Applications of sensor networks can be found in battlefield surveillance, environmental monitoring, biological detection, smart spaces, industrial diagnostics, etc. Although the technique of sensor networks has a very promising future, many challenges are still deserving lots of research efforts for its successful applications.

This book is devoted to coverage control, one of the most fundamental and important research issues in sensor networks. The aim of the book is to provide tutorial-like and up-to-date reference resources on various coverage control problems in sensor networks, a hot topic that has been intensively researched in recent years. Due to some unique characteristics of sensor networks such as energy constraint and ad-hoc topology, the coverage problems in sensor networks have many new scenarios and features that entitle them an important research issue in recent years. I have done my best to include in the book the most recent advances, techniques, protocols, results, and findings in this field. While I have tried to be as exhaustive as I could, the reader is advised to note that what is reported in this book is a picture of research approaches in this field taken at the middle of 2009.

Audience

This book is intended for graduate students, academic researchers, and industrial professionals who are interested in acquiring a big picture of various coverage control problems in sensor networks, including the problem scenarios, their assumptions and challenges, solution techniques and protocols. The book can serve as a reference book for undergraduate and graduate classes, or as a handbook for researchers, engineers, and developers working in the field of sensor networks.

Book Overview

This book is divided into four parts. The first part of this book provides general introductions, and the rest three parts are each devoted to a category of coverage control problems. In this book, we classify coverage problems into three categories, based on the coverage type, namely, *point coverage problems* (Part II), *area coverage problems* (Part III), and *barrier coverage problems* (Part IV).
Part I presents introductions on sensor networks and coverage control in sensor networks.

- Chapter 1 gives a short introduction to sensors, sensor nodes, and sensor networks, briefly describing the functions and characteristics of sensors, the architecture and components of sensor nodes, and the scenarios and applications of sensor networks. This chapter also discusses sensor network challenges and key research issues.
- Chapter 2 summarizes sensor coverage models mostly used in the literature, mainly elaborating their motivations, definitions, and applications. The sensor coverage model serves as a cornerstone of network-wide coverage control.
- Chapter 3 provides a big picture of various network coverage control problems, including the motivations, objectives, and design issues. We also discuss how the coverage control service can be integrated into the network protocol stack. At the end the chapter, we provide an informal definition and taxonomy for network-wide coverage control.

Part II is devoted to the point coverage problems. In the point coverage problem, the subject to be covered is a set of discrete points.

- Chapter 4 studies the node placement optimization problem for coverage configuration before network deployment, where the objective is to find the optimal locations to place sensor nodes to minimize network cost.
- Chapter 5 investigates the coverage lifetime maximization problem by controlling coverage characteristics in a randomly deployed network, where the objective is to optimally schedule sensors activities in order to extend network lifetime.

Part III is dedicated to the area coverage problems. In the area coverage problem, the subject to be covered is the whole sensor field.

- Chapter 6 discusses the *critical sensor density* (CSD) problem for coverage configuration before network deployment, where the objective is to find the least number of sensor nodes per unit area to provide complete coverage for the whole sensor field.
- Chapter 7 looks into the sensor activity scheduling problem of controlling network coverage characteristics in a randomly deployed network, where the objectives are to identify coverage redundant sensors and schedule sensors' activity in order to prolong the network lifetime.
- Chapter 8 introduces the node movement strategy problem for sensor networks containing mobile nodes, where the objective is to leverage mobile nodes to control network coverage. Mobile nodes change network coverage characteristics via

moving to the desired locations. The design of node movement strategy should balance between network coverage and movement cost.

Part IV discusses the barrier coverage problems. In the barrier coverage problem, the objective is to identify the desired coverage characteristics, if it exists, for a sensor network.

- Chapter 9 examines the coverage problems of building *intrusion barriers* for detecting intrusions of a mobile object when it traverses from one side to the other side of the sensor field. The trajectory of an intrusion mobile object is called its *traverse path*. The objective is to enable the covered points to form an intrusion barrier, stretching across the sensor field, and intersecting with every potential traverse path.
- Chapter 10 reviews the coverage problems of finding *penetration paths*. A penetration path is a continuous curve with arbitrary shape, spanning from one side to the other side of a sensor field. We assign a coverage measure (a real value) to represent the coverage characteristics of a single space point. The objective is to identify such a penetration path on which every single point satisfies the required coverage measure.

Acknowledgments

I would like to thank all the staff of Springer (Wayne Wheeler, Simon Rees, Catherine Brett) for their supports and assistances during the writing and production of the book. I am also grateful to my supervisors and colleagues who supported me and shared with me the exciting task of researching coverage issues in these years: Chua Kee Chaing, Vikram Srinivasan, Lim Hock Beng, Wang Wei, and Zhao Qun. Finally, I want to acknowledge my wife, Minghua, for her patience and support during the production of this book.

About the Author

Bang Wang received his Bachelor of Engineering and Master of Engineering degree from the Department of Electronics and Information Engineering in Huazhong University of Science and Technology (HUST) Wuhan, China in 1996 and 2000, respectively, and his PhD degree in Electrical and Computer Engineering (ECE) Department of National University of Singapore (NUS) Singapore in 2004. During his research career from 2004 to 2006, Dr. Bang Wang had worked as a research scientist in Nokia R&D center in Aalborg Denmark, as a research fellow in NUS Singapore, and as a research engineer in Panasonic Singapore Laboratories Singapore. Since 2007, Dr. Bang Wang has worked as a researcher in the Intelligent Systems Center (IntelliSys) in Singapore, an applied research center jointly set up by Singapore Technologies Engineering (ST Engineering) and Nanyang Technological

University (NTU). His main research interests include coverage and topology issues, distributed signal processing, resource allocation and optimization algorithms in wireless networks. Dr. Bang Wang had published more than forty technical articles in international journals, conferences, and books. He had also published one textbook and filed one patent. Dr. Bang Wang is a member of IEEE and has served as a technical program committee member for many international conferences.

Singapore *Bang Wang*
王 邦

Contents

Acronyms

2D	Two-dimensional
3D	Three-dimensional
ADC	Analogue-to-digital converter
ASIC	Application specific integrated circuit
BLUE	Best linear unbiased estimator
CDF	Cumulative density function
CPU	Central processor unit
CSD	Critical sensor density
DSC	Disjoint set cover
EEPROM	Electrically-erasable programmable read-only memory
FOV	Field of view
ILP	Integer linear programming
LLC	Logical link control
LP	Linear programming
MAC	Media access control
MEMS	Micro-electro-mechanical systems
OS	Operating system
OSI	Open system interconnection
RAM	Random access memory
RIS	Random independent sleeping
RMS	Root-mean-square
ROM	Read-only memory
RSIC	Reduced instruction set computer
TDMA	Time-division multiple access

Part I
Introduction

Chapter 1
Introduction

Abstract Sensors are devices that convert physical stimulus into recordable signals. Sensors have facilitated people to understand, monitor, and control machines and environments for many decades. A sensor node consists of not only sensor unit but also microcontroller unit, communication unit, storage unit, and power supply for producing, collecting, storing, processing, and delivering sensory data. The size and cost of a single sensor node has been reducing with the continuous advances of micro-electro-mechanical systems (MEMS) techniques. The miniaturization of sensor nodes has promoted the emergence of sensor networks, which normally consists of a large number of sensor nodes collaborating to accomplish advanced tasks. Applications of sensor networks are in a wide range, including battlefield surveillance, environmental monitoring, biological detection, smart space, industrial diagnostics, etc. Despite promising applications, there are also great challenges in designing, implementing, and operating sensor networks. Many research issues have been studied, and many solution approaches have been proposed for sensor networks. In this chapter, we provide some backgrounds and introduction about sensors, sensor nodes, and sensor networks.

1.1 Sensors

Generally speaking, a sensor is a device which responds to physical stimulus (such as heat, light, sound, pressure, magnetism, etc.) and converts the quantity or parameter of a physical stimulus into recordable signals (such as electrical signals, mechanical signals, etc.) [28]. These signals are normally digitalized to produce *sensing data*. Sensors represent a part of interfaces between the physical world and the world of electrical devices, such as computers, and facilitate people to understand, monitor, and control machines and environments.

Sensors come in many types and shapes and can measure almost all kinds of physical stimulus. Conventionally, sensors can be classified into different types according to the quantity to be measured [7, 28].

B. Wang, *Coverage Control in Sensor Networks,*
Computer Communications and Networks,
DOI 10.1007/978-1-84996-059-5_1, © Springer-Verlag London Limited 2010

- *Mechanical* quantities, such as position, displacement, velocity, accelerate, vibration, force, torque, pressure, etc.
- *Fluid* quantities, such as density, viscosity, thickness, surface morphology, etc.
- *Electrical* quantities, such as voltage, current, impedance, resistance, conductance, electric field, magnetic field, polarization, etc.
- *Thermal* quantities, such as temperature, IR imaging, etc.
- *Chemical* quantities, such as concentration, PH values, enzymes, ions, etc.
- *Biological* quantities, such as concentration of enzyme substrates, antigens, antibodies, etc.

Sensors of the same type can be compared by their performance characteristics. Some commonly used sensor characteristics are as follows [28].

- *Transfer function* shows the functional relation between physical input stimulus and electrical output signal.
- *Sensitivity* describes the changes of the output signal relative to the change of the input stimulus.
- *Dynamic range* specifies the range of the input physical stimuli that can be converted into electrical signals.
- *Accuracy* is defined as the largest expected error between actual and ideal output signals.
- *Hysteresis* is the range of the expected error in terms of the measured quantity.
- *Resolution* is defined as the minimum detectable signal fluctuation.
- *Noise* exists in the output signal in addition to the converted signal of physical stimulus. Many noises come from the electronic circuits. In some cases, the noise of the sensor is less than the fluctuations in the physical signal and may be neglected. In many other cases, noises cannot be neglected, which limit the performance of a sensor-based system. Noise is generally distributed across the frequency spectrum. Many common noise sources produce a white noise distribution, where the spectral noise density is the same at all frequencies.

A typical sensor consists of excitement elements and electronic circuits. The excitement element does not directly produce voltage or current changes but rather reacts to physical stimulus by changing its resistance, capacitance, or inductance. Such changes are captured by sensor circuits to produce voltage changes as output electrical signals. The sensor electronics limit the performance, cost, and range of applicability of a sensor. If carried out properly, the design of the sensor electronics can allow the optimal extraction of information from a noisy signal.

Example The operational principle of an acoustic sensor is illustrated by Fig. 1.1. The acoustic sensor is a condenser microphone, which converts sound signals to electrical signals. The flexible diaphragm and the fixed back plate consist of a parallel-plate capacitor. As sound waves hit the condenser microphone, the diaphragm vibrates in response to sound pressure, and the distance between the flexible diaphragm and the fixed back plate changes, which further causes the change of the plate electrical charge and produces the output electrical signals. In Fig. 1.1, the electronic circuit is a resistor–capacitor circuit and provides a constant charge

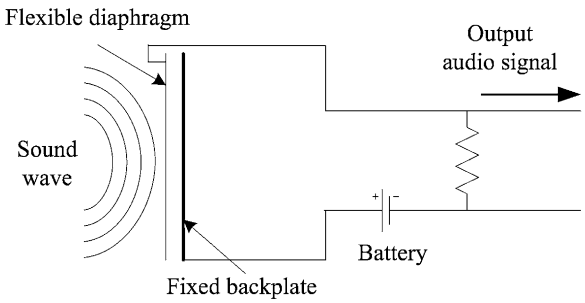

Fig. 1.1 An acoustic sensor–condenser microphone

Q to the capacitor. The capacitance of the capacitor is inversely proportional to the distance between the two plates. The capacitance equation is $C = \frac{Q}{V}$, where Q is the charge in coulombs, C is the capacitance in farads, and V is the potential difference in volts. When the capacitance C changes due to the vibration of the diaphragm, the voltage across the capacitor changes, and the voltage change is seen across the series resistor. The voltage across the resistor is amplified for further processing or recording.

With the advances of micro-electro-mechanical system (MEMS) technology, the size of a microphone sensor can be made very small. For example, the AKU1126 microphone by Akustica [1] Sensory Silicon^{TM} is a CMOS MEMS microphone chip with integrated acoustic transducer and analog output amplifier, and with the package size of only 2 mm $\times$ 2 mm $\times$ 1.25 mm.

1.2 Sensor Nodes

A sensor node, which is the basic unit of a sensor network, not only integrates sensors for sensing the physical world but also includes other units to process and deliver sensing data. A typical architecture of a sensor node is shown in Fig. 1.2, which consists of *sensor unit*, *communication unit*, *microcontroller unit*, and *memory* and *power unit*. A sensor node may also include some other units, depending on application scenarios and requirements, such as a locomotive unit that enables a sensor node to move around, an energy scavenge unit that obtains energy from the physical world, a GPS unit that acquires node geographical location, etc. In what follows, we briefly introduce the functionalities of each component shown in Fig. 1.2 and refer the interested reader to [2, 16, 24] for more discussions.

Microcontroller Unit The microcontroller is the core of a sensor node. Its functionalities include collecting and processing the sensing data from the sensor unit, deciding where and when to send the data, and controlling data reception from other sensor nodes, etc. We may liken the microcontroller to the *central processor unit* (CPU) of a desktop computer; however, a microcontroller consumes much less energy compared with a CPU. A microcontroller can be implemented by an

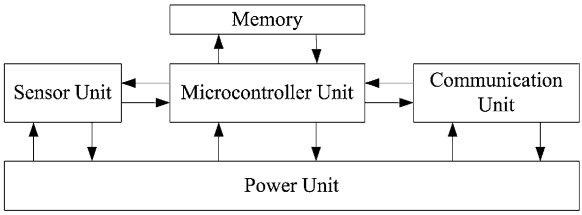

Fig. 1.2 A typical sensor node hardware architecture

application-specific integrated circuit (ASIC) which is designed for particular applications. A microcontroller can also be implemented by applying general *reduced instruction set computer* (RSIC) architectures running on some *operating systems* (OS), such as the famous TinyOS [26]. TinyOS is an open-source operating system particularly designed to meet the severe constraints—small memory and low power—of sensor nodes. Its event-driven execution model allows the scheduling flexibility, which is essential to a wireless sensor node due to the unpredictable nature of physical world stimuli and wireless communications.

Usually, a microcontroller can operate in different modes in order to save energy. This is done by only switching on some parts of the microcontroller and switching off the other parts. Some general operational modes include normal *active* mode, *idle* mode, and *sleep* mode. In the active model, all parts of the processor are fully powered. In the idle model, clocks and peripherals are active, and any interrupt will cause the microcontroller return back to the active mode. In the sleep mode, only the real-time clock remains active, and a wake-up occurs every fixed time interval.

There are several commercial microcontroller products in the market. For example, the SA-1100^{TM} model of the Intel StrongARM is a fairly high-end processor for hand-held devices like PDAs. It has a 32-bit RISC core and can run at up to 206 MHz [15]. The power consumption of Intel StrongARM is up to 400 mW in the active mode, up to 100 mW in the idle mode, and up to 50 µW in the sleep mode. The MSP 430^{TM} from Texas Instruments [14] is explicitly intended for embedded applications. It runs a 16-bit RISC core at considerably lower clock frequencies (up to 4 MHz). The power consumption of MSP430 is about 1.2 mW in active mode and about 0.3 µW in the deep sleep mode. The ATmega128A^{TM} from Atmel Corporation [5] is an 8-bit microcontroller, also intended for embedded applications with low power consumption. The power consumption of ATmega128 varies between 6 mW and 15 mW in idle and active modes, and is about 75 µW in power down mode.

Memory The memory component includes both the on-chip *random access memory* (RAM) used by the microcontroller and the on-board *read-only memory* (ROM) used for storing program codes and sensory data. While RAM is fast, its main disadvantage is that it loses its content if power supply is interrupted. The typical ROM includes Electrically Erasable Programmable ROM (EEPROM) and flash memory, while the later is similar to EEPROM but allowing data to be erased or written in blocks instead of only a byte at a time. The volume and size of memory unit are much dependent on the application requirements, ranging from hundreds of KB to

hundreds of MB. The volume of a memory of the same size has being continuously increased due to recent technique advances.

The power consumption of on-chip memory is typically included in the power consumption numbers given for the microcontroller. Reading from and writing to on-board flash memory consumes time and energy. The reading times and read energy consumption are similar between different types of flash memory. But there are considerable differences in writing times and write energy consumption among different types of memory. As an example, the power consumption rate for reading from and writing to the flash memory in a Mica sensor node [20] are 1.111 nAh and 83.333 nAh, respectively.

Sensor Unit We have introduced sensors in the previous section. A sensor unit may consist of more than one type of sensors. Furthermore, a sensor unit usually includes analogue-to-digital converters (ADCs), which convert the analog signals produced by the sensors to digital signals and input to the microcontroller for further processing. It is widely assumed in the literature that each sensor has a certain sensing capability that can *cover* some area such that the quantity of interest in this area can be reliably and accurately sensed. In other words, an *active* sensor can cover some subject when it is turning on for sensing. The sensor unit can also be turned off for not sensing, and hence it does not cover any subject at all. In such a case, the sensor unit is deactivated, or it is called in a *sleep* state. The activation and deactivation of the sensor unit are controlled by the microcontroller according to the implemented coverage control algorithms in each sensor node. This book is devoted to various coverage control problems in sensor networks and concerns with how to efficiently control the activation and deactivation of sensor nodes in a network-wide way.

The power consumption rates are much different for different types and makes of sensors. The power consumption of a sensor also depends on the application-specific sampling rate and accuracy requirement. The sampling rate determines how frequently to output a unit of sensing data; and the accuracy requirement determines how to enforce analogue-to-digital conversion and quantization. In most cases, a sensor does not consume any energy if it is in the sleep state. Furthermore, the power consumption rate for producing one-bit sensing data by an active sensor is usually regarded as much less than that for transmitting or receiving one-bit sensing data by a transceiver. However, it is still important to enforce some control of the sensor unit for more overall energy savings, since processing and transmitting the sensing data produced by a sensor unit also consume energy. Finally, we provide an example: For the humidity sensor of the Mica Weather Board [20], the sampling rate is 500 Hz, and the current draw of the sampling is 0.775 mA.

Communication Unit The communication unit is used to exchange data among individual nodes. Both wired and wireless communications are allowable for sensor nodes. In the case of wired communications, sensor nodes have to be connected via wires or field buses, which limits the flexibility and scalability of a sensor network. In both industrial and academic domain, wireless communication is of more

interests. In the case of wireless communications, sensor nodes are equipped with transceivers and can easily form an ad-hoc *wireless sensor network* (WSN) in an unattended way. A transceiver which consists of a transmitter, a receiver, and all other necessary circuitry such as modulator and demodulator, power amplifiers, filters, antenna, etc., converts bit streams to or from radio waves. A radio transceiver usually works in a half-duplex way, since transmitting and receiving at the same time over the wireless media is impractical in most cases.

Many transceivers can operate in four different operational modes, namely, *transmit*, *receive*, *idle*, and *sleep* mode. In the transmit mode, the transmit part of the transceiver is active, and the antenna radiates energy. In the receive mode, the receiver part is active. In the idle mode, a transceiver is ready to receive but is not currently receiving anything. In the idle mode, only a part of the receive circuitry is active, and the rest can be switched off. In the sleep mode, most parts of a transceiver are switched off. Usually, the sleep mode is the most energy saving mode with the least energy consumption rate, and the next one is the idle mode. Some transceivers offer additional submodes that turn different parts of the transceiver on and off for further energy saving. However, a considerable amount of energy is consumed when a transceiver transits from one state to another. The mode of the transceiver is controlled by the microcontroller according to the implemented algorithms and protocols.

There are lots of commercial transceivers designed for wireless sensor networks (WSNs). They normally work on the Industrial, Scientific, and medical (ISM) frequency band which gives free radio, huge spectrum allocation and global availability. The transceivers for WSNs are all single-chip solutions with a fairly low power consumption rate. For example, the ZMN transceiver family™ from RFM [21] works on the 2.4 GHz, with up to 250 kbps data rate. Take the module ZMN2405 [22] for example; its transmit power is up to 1 mW, and the current draw in the receive, transmit, and sleep mode is 27 mA, 28 mA, and 3 μA, respectively. Another example is the transceiver MPR2400™ [11], which is integrated within the MICAZ sensor node by Crossbow [29]. The typical current draw is 17.4 mA, 19.7 mA, 20 μA, and 1 μA for the transmit, receive, idle, and sleep mode, respectively.

Power Unit The power unit of a sensor node usually consists of one or more nonrechargeable batteries, Sometimes, chargeable batteries are used if some energy scavenging device (such as photovoltaic cells) is also present on the node. Both nonchargeable and chargeable batteries output power by converting the stored chemical energy to electrical energy. Different types of batteries may have different capacities. It is desirable that a battery has high capacity (milliampere-hour, mAh) at small size, tiny weight, and low cost. For example, the voltage of a typical AA battery is 1.5 V, and its capacity is about 2890 mAh. Such an AA battery is often of a cylinder shape with diameter 14.5 mm and height 23 mm, and its weight is about 23 g.

Example A sensor node is implemented by a sensor board which integrates all the aforementioned components and other necessary circuitry. Many actual sen-

Fig. 1.3 Photograph of a MICAz sensor node (Courtesy of Crossbow [29])

sor nodes have been manufactured for research and development of wireless sensor networks (see, e.g., [6, 13] for summaries of sensor node developments). Figure 1.3 presents a photograph of a MICAz sensor node by Crossbow Corporation [29]. The top layer of the node is the sensor board MPR2400^{TM} [11], of 58 mm × 32 mm × 7 mm large and of 18 g weight. The microcontroller is based on the Atmel ATmega128L, and three memory types are used, namely, program flash memory of 128 KB, configuration EEPROM 4 KB, and measurement flash 512 KB. The sensor board can support many sensor types, including acoustic, light, magnetometer, and accelerometer sensor, and contains a 10-bit ADC. A MICAz node runs on 2.4 GHz at up to 250 kbps data rate and conforms with IEEE 802.15.4 standard. The bottom is a battery case which can include two AA batteries. The current draw of the microcontroller in active mode is about 8 mA and in sleep mode is less than 15 μA. The current draw of the transceiver is 19.7 mA in receive mode, up to 17.4 mA in transmit mode, 20 μA in idle mode (with voltage regulator on), and 1 μA in sleep mode (with voltage regulator off).

1.3 Sensor Networks

Sensor nodes are usually deployed in a field of interests to monitor some physical phenomena. Such a field is called a *sensor field*, and the sensor nodes form a *sensor network*. Diverse scenarios exist in the architecture and management of sensor networks. Numerous sensor network applications are expected to emerge in the near future.

1.3.1 Sensor Network Scenarios

A sensor network normally consists of a large number of sensor nodes and one or more sinks. Sensor nodes monitor physical phenomena and produce sensory data. A *sink*, on the other hand, does not generate any data by itself but collects data from sensor nodes. A sink can be regarded as a gateway between a sensor network and

other networks, or an interface between a sensor network and the people operating the sensor network. A sink is often assumed as resource abundant, without energy supply limit and with advanced computation capability. Aside from data collection, a sink is also a central controller which can execute network management algorithms and instruct sensor nodes only the computation results.

Sensor networks may work in different architectural and operational scenarios, depending on the sensor nodes' and sinks' capabilities and on the communication paradigm used by the sensor nodes and sinks. For example, some sensor nodes may have more advanced sensor unit, microcontroller, or radio transceivers, which enable them to take more responsibilities in the whole sensor network. Sensor nodes or sinks may also be equipped with locomotives which enable them to move around the sensor field for better performing tasks. In what follows, we introduce several typical sensor network scenarios.

Homogeneous vs. Heterogenous Networks In a homogeneous sensor network, all sensor nodes have the same sensing, processing, communication, and other capabilities (excluding the initial power supply). Figures 1.4(b) and (d) illustrate two homogeneous networks. In a heterogeneous sensor network, sensor nodes have different capabilities. For example, a node may have a stronger sensor unit and can cover a larger area. Figures 1.4(a) and (c) present two examples of heterogeneous networks. Two nodes that have different amount of initial power supplies but with identical all the other capabilities are usually not considered as two heterogenous nodes. A homogeneous network can be considered as just many copies of a sensor node in different geographical locations, and due to such characteristics, many theoretical analyses are carried out for homogeneous networks. On the other hand, it is generally recognized that the performance of a homogeneous network can be greatly improved by adding some more advanced sensor nodes or some mobile sensor nodes. For example, in Fig. 1.4(c) some advanced sensor nodes serve as cluster heads to form clusters.

Stationary vs. Mobile Networks In a stationary sensor network, all sensor nodes are fixed and cannot move around after they have been deployed. A mobile sensor node is equipped with a locomotive unit and can move around after deployment. In general, a mobile node is more expensive than its stationary compeer. It is usually considered in the literature that a mobile sensor network is a network consisting of only mobile nodes, and a hybrid sensor network is a network consisting of both stationary nodes and mobile nodes. As mobile nodes can move to desired locations, it is not unexpected that using mobile nodes can improve sensor network performance. Such performance improvements are often at the cost of expensive mobile nodes and more energy consumed for moving. In some cases, a sink can also be a mobile node, and it moves around to collect sensing data. For example, Fig. 1.4(d) illustrates a trajectory along which a mobile sink moves.

Single-Hop vs. Multi-Hop Networks There are two basic communication paradigms between sensor nodes and sinks, namely, *single-hop* communication and

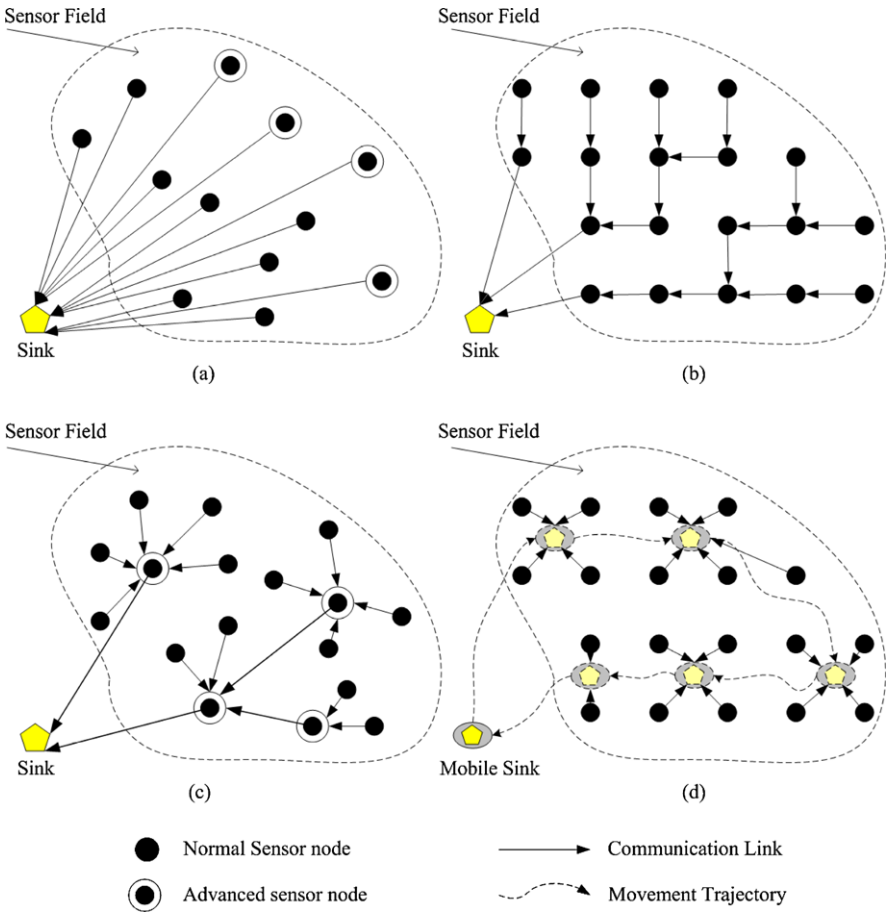

Fig. 1.4 Sensor network scenarios: (**a**) single-hop, heterogenous, stationary network; (**b**) multi-hop, homogeneous, stationary network; (**c**) multi-hop, heterogeneous, stationary network; (**d**) single-hop, homogeneous, stationary network with a mobile sink

multi-hop communication. In a single-hop sensor network, all sensor nodes transmit directly to the sink. Figure 1.4(a) illustrates a single-hop sensor network where all stationary sensor nodes transmit directly, via either wired or wireless communications, to the sink outside the sensor field. Such a direct communication paradigm might be very expensive, especially for a large-scale network. If wired communication is used, the cost of wires might be much larger than that of sensor nodes. If wireless communication is used, the long distance between a node and the sink requires very high transmission power. Another option is to use mobile sinks which move around in the sensor field to collect data via short-distance radio communication, as illustrated by Fig. 1.4(b). In such a case, a sensor node only transmits to the sink when the sink moves close to itself, and due to the reduced transmission distance, a considerable amount of transmission energy can be saved.

In a multi-hop sensor network, instead of transmitting to the sink directly, some sensor nodes use a multi-hop path consisting of other nodes as relays to deliver their data to the sink. Many routing algorithms, such as the Dijkstra shortest-path algorithm [10], can be used to find an appropriate multi-hop path for each node. Figure 1.4(b) illustrates a flat multi-hop sensor network where only three nodes closer to the sink transmit to the sink directly, and others uses multi-hop communications. Figure 1.4(c) illustrates a hierarchical multi-hop sensor network where clusters are formed, and some powerful nodes serve as cluster heads and other nodes serve as cluster members. In a cluster, each cluster member transmits to its cluster head via either single-hop or multi-hop transmission.

The choice of a single-hop or a multi-hop network is heavily dependent on the application requirements. In general, a single-hop network is easy to maintain and suitable for small-scale networks. Multi-hop networks, on the other hand, are often the practical choices for large-scale networks with hundreds or thousands of nodes. It is worth noting that, unlike the traditional store-and-forward networks, in-network data processing is allowed and even encouraged in sensor networks. In the store-and-forward networks, an intermediate relay node does not attempt to modify received data but just retransmit the data to other nodes. However, the idea of in-network data processing is to allow a node to process its received data packets, analyzing the packets' content and even aggregating them to a new packet yet with reduced volume. Multi-hop networks, obviously, are the better choice for in-network processing.

1.3.2 Sensor Network Applications

As almost all kinds of physical phenomena can be sensed by different types of sensors, it is not surprised to find applications of sensor networks in almost any environment. Application domains of sensor networks include military, environmental, industrial, home, medical applications, etc. [2, 9, 16, 24, 32]. In what follows, we present a brief introduction and example applications of sensor networks.

Military Applications Sensor networks can be integrated into the military command, control, communications, and computing (C4) systems. Wireless sensor networks, which can be rapidly scattered to critical terrains, routes, paths, and straits, can provide battlefield intelligence, such as the location, numbers, movement, and identity of troops and vehicles, and can be used to detect, classify, and track the trajectory of enemy vehicles.

Some example military projects of sensor networks include using sensor networks to detect and track vehicles [4], using sensor networks to detect and locate a sniper shoot [23].

Environmental Applications Some environmental applications of sensor networks include tracking the habitat of birds and animals, observing bio-diversity and

bio-complexity, monitoring environmental conditions for precision agriculture, detecting disasters such as forest fire detection and flood detection, studying air pollution, etc.

There are lots of example projects of using sensor networks for environmental monitoring. For example, a sensor network has been deployed in Yosemite National Park for meteorology and hydrology study [19]. A sensor network has been deployed in the Volcán Reventador Volcano for studying volcanoes' interior structure [27]. A sensor network has been deployed to monitor the habitat of the nesting petrels on Great Duck Island [20].

Industrial Applications Sensor networks can also find many industrial applications, such as manufacture automation, warehouses and products management, automated reading for gas, water, and electric meters, lighting control, smoke and nocuous gas detection, etc. Sensor networks can also be used to monitor the structural health of building, bridges, and roads and to check the physical condition of water and gas pipes. At present, structures are monitored primarily through manual inspections. Untethered sensors can help to automate the monitoring process by providing timely and detailed information about incipient cracks or about other structural damage.

Some example industrial projects include the Pipenet project which uses sensor network to monitor public water supply pipes [25], the Wisden project which monitors the structural health [30], and the project that deploys a sensor network to monitor the structural health for the Golden Gate Bridge [17].

Home Applications Sensor networks can help to create a smart environment. With the advances of technology, tiny yet smart sensors can be buried in appliances, such as vacuum cleaners, micro-wave ovens, and refrigerators. These sensors inside the domestic devices can interact with each other and with external networks such as the Internet. This allows users to manage their devices locally and remotely.

An example of smart environment is the "Aware Home" at Georgia Institute of Technology [12] with the aim of developing infrastructure for ubiquitous sensing and recognition of activities in environments. Another example is the Smart kindergarten project [8], which uses WSNs to create a smart environment for early childhood education.

Medical Applications Physical sensors and bio-sensors can monitor body temperature, pulse rate, perspiration, oxygen and glucose level in blood, and other physical and biochemical parameters. These sensors can be wear in clothes or even implanted into human bodies. A wireless body sensor network integrates these wearable or implanted sensor nodes into a network and connects them with external networks for monitoring patients with chronic disease, hospital patients, and elderly patients [31].

An example project is the CodeBlue which develops a sensor network for monitoring and tracking of patients [18]. Another example project is the automatic dietary monitoring system which applies a body sensor network to facilitate continuous eating-behavior report and analysis and to provide dietary suggestions [3].

1.4 Challenges and Issues

While numerous applications of sensor networks are expected to emerge in almost every domain, many new research and development challenges must be studied and settled for the successful realization of sensor networks. As a conclusion of this chapter, we outline some key research issues, among which the coverage problems are the theme of this book.

1.4.1 Sensor Network Challenges

Sensor networks are expected to have very promising applications in almost every domain; there are also many technical challenges that still need considerable research and development efforts. Many of these challenges that are specific to sensor networks come from the severe constraints of limited resources and capabilities of individual sensor nodes. On the other hand, sensor networks also face some challenges common to many other types of networks. In what follows, we briefly discuss the most important challenges in sensor networks.

Energy Efficiency In sensor networks, it is of crucial importance to consume nodes' energy wisely and efficiently. As sensor nodes are normally equipped with nonchargeable batteries with limited energy supply, a sensor network cannot function well after a fraction of nodes run out of energy. In other words, the effective operational time of a network, or the network lifetime, is limited. Many energy-conserving mechanisms have been implemented in the node hardware such as using different operational modes, each with different energy consumption rate in node hardware. The hardware activity scheduling, however, should be coordinated and enforced in a network-wide manner such that the network performance will not be sacrificed yet nodes' energy can be saved.

Network Autonomy Sensor nodes can be either deterministic placed or randomly scattered into a field of interests. In a remote or dangerous field, randomly scattering nodes might be the only way to deploy a sensor network. In such cases, the untended nodes should self-organize into an autonomous network to decide the structure and topology of the network. Such an autonomous network should be able to schedule sensing tasks and to arrange delivery routes all by itself. It is also required that an autonomous sensor network should be able to monitor its own health and status and to adapt its operational parameters in different situations. Sometimes, a sensor network should also be able to interact with external maintenance mechanisms or external network interfaces for better operation.

Network Scalability In many envisioned applications, the number of sensor nodes in a sensor network may be in the order of thousands, tens of thousands, or even millions. In such large-scale networks, scalability is a critical factor, guaranteeing that the network performance does not significantly degrade as the network

size increases. Algorithms and protocols designed for small-scale networks do not work necessarily well in large-scale ones. To achieve network scalability, distributed and localized algorithms are preferred. Also those protocols with few message overheads and little resource requirements are easy to scale to large-scale networks.

Fault Tolerance Sensor nodes are cheap devices and prone to failures. Node failure rate may be very high if they are deployed in hostile or harsh environments. Fault tolerance, or robustness, should be included in the design and implementation of algorithms for sensor networks such that the network performance is not sensitive to individual node failures. It is also desirable that the network performance can degrade as gracefully as possible with respect to node failures.

Data Accuracy Obtaining accurate information is the main task of sensor networks. Accuracy can be improved through joint signal processing by cooperative sensors. However, in many cases maintaining high information accuracy is in conflict of extending long network lifetime, since more sensors are used for sensing, processing, and delivering tasks. It is therefore desirable to make a good balance between data accuracy and energy consumption.

Information Security Information security, which is a basic yet common requirement in almost all types of networks, requires that sensing data should be accessed, transmitted, and processed securely and privately. However, security algorithms are usually resource demanding and should be modified to adapt to resource-constraint sensor networks.

1.4.2 Key Research Issues

On the road of successful applications, many research issues need to be first studied for sensor networks. Our view of the key research issues in sensor networks, which should be regarded as only one of many possible solutions, can be broadly classified into three groups, namely, *node system*, *middleware services*, and *communication protocols*, as shown in Fig. 1.5. In what follows, we simply outline some key research issues.

Node System Each sensor node is a self-contained system. There are lots of research tasks in the design, implementation, and improvement of the hardware platforms, including those individual components like microcontroller, sensor unit, and the sensor board that integrates these components. The operating system, the database technology, and other system supporting software are also worth of considerable research efforts.

Communication Protocols The communication protocols enable sensor nodes to exchange information with each other and to transmit sensing data back to the sink.

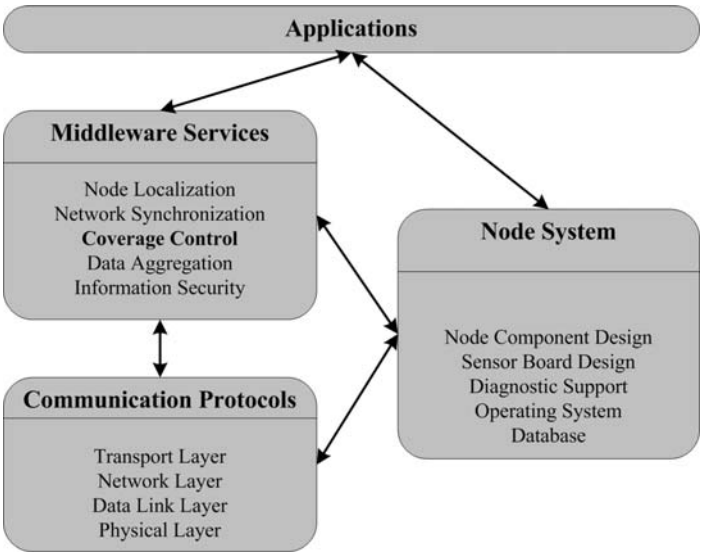

Fig. 1.5 A categorization of key research issues in sensor networks

The lower four layers of the communication protocol stack are the physical layer, data link layer, network layer, and transport layer. Due to the energy constraint and other hardware constraint, most of these protocols designed for other networks may not be directly suitable for sensor networks. Many research efforts have been devoted to the design and implementation of appropriate communication protocols.

Middleware Services Middleware services are software modules used to facilitate network configuration and management and to improve network efficiency and application performance. It is called middleware as it is considered to be below the application layer and above communication protocol layer. Some commonly researched middleware services include node localization service, network synchronization service, *coverage control* service, data aggregation service, and information security service, etc. For example, node localization service is to help each node to find its coordinates, and network synchronization service is to synchronize nodes' internal clocks.

The coverage control problem is the theme of this book, which is viewed as one of the middleware services. In a single node, it is to control the activation and deactivation of the sensor unit. However, the coverage control middleware should be designed from a network-wide perspective. This is to coordinate different nodes' sensing tasks such that individual nodes' energy can be saved while the network-wide sensing quality can be guaranteed. There are also other types of coverage control problems, such as finding coverage characteristics of a sensor network, deploying sensor nodes for guaranteeing network coverage, etc. We will study various coverage control problems in this book. Before that, we first introduce the concept of sensor coverage model in the next chapter.

References

1. Akustica: http://www.akustica.com/
2. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: A survey. Computer Networks **39**(4), 393–422 (2002)
3. Amft, O., Tröster, G.: On-body sensing solutions for automatic dietary monitoring. IEEE Pervasive Computing **8**(2), 62–70 (2009)
4. Arora, A., Dutta, P., Bapat, S., Kulathumani, V., Zhang, H., Naik, V., Mittal, V., Cao, H., Demirbas, M., Gouda, M., Choi, Y., Herman, T., Kulkarni, S., Arumugam, U., Nesterenko, M., Vora, A., Miyashita, M.: A line in the sand: A wireless sensor network for target detection, classification, and tracking. Computer Networks (Elsevier) **46**(5), 605–634 (2004)
5. Atmel Corporation: http://www.atmel.com
6. Bokareva, T.: Mini hardware survey. http://www.cse.unsw.edu.au/~sensar/hardware/hardware_survey.html (2009)
7. Brignell, J., White, N.: Intelligent Sensor Systems. Institute of Physics Publishers, Philadelphia (1996)
8. Chen, A., Muntz, R.R., Yuen, S., Locher, I., Park, S.I., Srivastava, M.B.: A support infrastructure for the smart kindergarten. IEEE Pervasive Computing **1**(2), 49–57 (2002)
9. Chong, C.Y., Kumar, S.P.: Sensor networks: Evolution, opportunities, and challenges. Proceedings of the IEEE **91**(8), 1247–1256 (2003)
10. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd Edition. MIT Press, Cambridge (2001)
11. Crossbow: Micaz data sheet. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf (2009)
12. Essa, I.A.: Ubiquitous sensing for smart and aware environments. IEEE Personal Communications **7**(5), 47–49 (2000)
13. Hill, J., Horton, M., Kling, R., Krishnamurthy, L.: The platform enabling wireless sensor networks. Communications of the ACM **47**(6), 41–46 (2004)
14. Texas Instruments: http://www.ti.com
15. Intel: Strongarm sa-1100 microprocessor data sheet. Intel Product Documentation (2000)
16. Karl, H., Willig, A.: Protocols and Architectures for Wireless Sensor Networks. Wiley, New York (2005)
17. Kim, S., Pakzad, S., Culler, D., Demmel, J., Fenves, G., Glaser, S., Turon, M.: Health monitoring of civil infrastructures using wireless sensor networks. In: The International Conference on Information Processing in Sensor Networks (IPSN), pp. 1–10 (2007)
18. Lorincz, K., Malan, D.J., Fulford-Jones, T.R., Nawoj, A., Clavel, A., Shnayder, V., Mainland, G., Welsh, M., Moulton, S.: Sensor networks for emergency response: Challenges and opportunities. IEEE Pervasive Computing **3**(4), 16–23 (2004)
19. Lundquist, J.D., Cayan, D.R., Dettinger, M.D.: Meteorology and hydrology in Yosemite national park: A sensor network application. In: International Conference on Information Processing in Sensor Networks (IPSN), pp. 518–528 (2003)
20. Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless sensor networks for habitat monitoring. In: ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), pp. 88–97 (2002)
21. RFM: http://www.rfm.com
22. RFM: Zmn2405–zigbee transceiver module. http://www.rfm.com/products/pdf/zmn2405.pdf (2009)
23. Simon, G., Maróti, M., Ákos Lédeczi, Balogh, G., Kusy, B., Nádas, A., Pap, G., Sallai, J., Frampton, K.: Sensor network-based counter sniper system. In: ACM the 2nd International Conference on Embedded Networked Sensor Systems (SenSys), pp. 1–12 (2004)
24. Sohraby, K., Minoli, D., Znati, T.: Wireless Sensor Networks: Technology, Protocols, and Applications. Wiley, New York (2007)
25. Stoianov, I., Nachman, L., Madden, S., Tokmouline, T.: Pipenet: A wireless sensor network for pipeline monitoring. In: The International Conference on Information Processing in Sensor Networks (IPSN), pp. 1–10 (2007)

26. TinyOS: http://www.tinyos.net
27. Werner-Allen, G., Lorincz, K., Welsh, M., Marcillo, O., Johnson, J., Ruiz, M., Lees, J.: Deploying a wireless sensor network on an active volcano. IEEE Internet Computing **10**(2), 18–25 (2006)
28. Wilson, J.S.: Sensor Technology Handbook. Elsevier, Amsterdam (2005)
29. Xbow: Crossbow technology inc. http://www.xbow.com/ (2009)
30. Xu, N., Rangwala, S., Chintalapaudi, K.K., Ganesan, D., Broad, A., Govindan, R., Estrin, D.: A wireless sensor network for structural monitoring. In: ACM International Conference on Embedded Networked Sensor Systems (SenSys), pp. 13–24 (2004)
31. Yang, G.Z.: Body Sensor Networks. Springer, Berlin (2006)
32. Yick, J., Mukherjee, B., Ghosal, D.: Wireless sensor network survey. Computer Networks (Elsevier) **52**(12), 2292–2330 (2008)

Chapter 2
Sensor Coverage Model

Abstract Sensor coverage models are used to reflect sensors' sensing capability and quality. They are abstraction models trying to quantify how well sensors can sense physical phenomena at some locations, or in other words, how well sensors can cover such locations. In almost all cases, sensor coverage models can be mathematically formulated as a coverage function of distances and angles. The inputs of such a coverage function are the distances between a particular space point and sensors' locations, and the output is a nonnegative real-valued number and is called *coverage measure* of this space point. In some cases, a space point is said to be covered if its coverage measure satisfies some predefined threshold. On the other hand, sensor types are diverse, and each sensor type has its own manner of sensing physical stimuli. Also application scenarios are various, and each application scenario has its own way of interpreting sensory data. As such, sensor coverage functions can be defined in different forms and are subject to different interpretations, depending on sensor types and application scenarios. This chapter introduces some common sensor coverage models, including their motivations, formulations, interpretations, and applications.

2.1 Motivations

A sensor converts physical stimuli into electrical or other recordable signals. These signals are further processed to output digital sensing data which are embedded with comprehensible information. An interesting and important question is: how well a sensor works? In order to address this question, many measurement mechanisms have been used to quantify and compare sensors. As introduced in the previous chapter, some sensor characteristics, such as transfer function, sensitivity, dynamic range, accuracy, etc., can be used to measure how well a sensor reacts to physical stimuli. In this chapter, we introduce *sensor coverage model* as another mechanism to measure sensors' sensing capability and quality. In what follows, we first discuss some motivations of sensor coverage models with examples.

B. Wang, *Coverage Control in Sensor Networks,*
Computer Communications and Networks,
DOI 10.1007/978-1-84996-059-5_2, © Springer-Verlag London Limited 2010

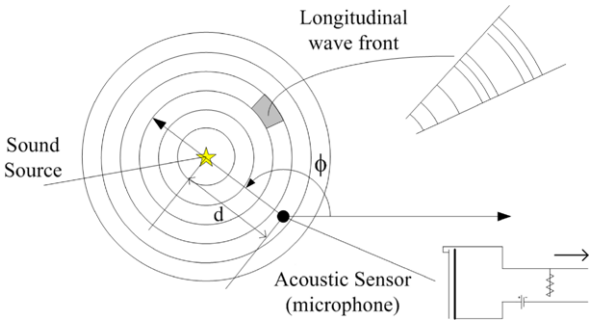

Fig. 2.1 Illustration of radiation of sound waves from a point source in free space

Example Let us first consider an example of using an acoustic sensor to measure sound pressure. Figure 2.1 illustrates the radiation of sound waves from a sound source in the free space. Suppose that the sound source emits a pure tone characterized by a sine function $a\sin(2\pi f + \theta)$, where a is the maximum amplitude, f the frequency, and θ the initial phase. The root-mean-square (RMS) amplitude $a_{\text{ams}} = \frac{a}{\sqrt{2}}$ is used to measure the sound pressure

$$L_p = 20\log_{10} a_{\text{ams}} + C \text{ (dB)},$$

where the constant C is the reference sound pressure (an internationally agreed value of C is 94 dB). A free space is a homogeneous medium, free from boundaries or reflecting surfaces. In such a free space, the sound waves radiated from a sound source will diffuse in all directions, and its amplitude (or energy in terms of a_{rms}) attenuates with distances. The sound pressure at a given point, at a distance d (in meters) from the source, can be computed as [7, 9, 16]

$$L_p = L_{\text{ref}} - 20\log_{10}\left(\frac{d}{d_{\text{ref}}}\right) \text{ (dB)},$$

where L_{ref} is the sound pressure at a reference point (usually greater than 1 meter to avoid source near field effects), and d_{ref} is the distance between the reference point and sound source. From this expression it can be seen that in the free space, the sound pressure decreases by 6 dB when the distance d doubles.

All kinds of acoustic sensors have limitations on the measurable sound pressure level (in dB), beyond which an accurate measurement cannot be obtained. The lower measurement limit of a microphone is established by its cartridge thermal noise [27]. There are two sources of thermal noise, air damping and preamplifier circuitry. The air damping causes a white noise that is a property of the microphone. The preamplifier has low-frequency noise which is inversely proportional to frequency and white noise. The thermal noise determines the lower measurement limit of an acoustic sensor, which is the dB level that would be read by a measurement instrument connected to the microphone output when there is no acoustic pressure applied to the microphone. If the distance between a sound source and an acoustic sensor is too large, the sound pressure at the sensor location is too small and cannot be accurately measured by the sensor. This suggests that a sensor may only sense

some object within a limited range. Or in other words, a sensor can cover some region with limited area.

Example Let us consider another example of using a thermometer sensor to measure environmental temperature. The thermocouple of a thermometer consists of two electrical conductors of dissimilar materials. One is called the measurement junction, and the other is called the reference junction with known reference temperature. When the temperature of the measurement junction is different from the reference junction, a current will flow in the circuitry with the intensity proportional to their temperature difference. Although the measurement is the temperature of the air around the thermometer, it may be extended to infer the temperature of some other space points not far away from the thermometer. Such inferences may not be very accurate; however, they are useful in practice. For example, if the measured temperature is 30°C, then we may infer that a space point with distance of 10 meters away from the thermometer has the same temperature. This suggests that the sensing data of a sensor can be applied to the space points not only around the sensor but also close to the sensor. Or in other words, a sensor can cover some space with limited area.

There are also some other types of sensors whose sensing capability and quality can be related to the distances between a space point and sensors. These motivate the use of *sensor coverage model*, as one of many other mechanisms, to model how well sensors can sense physical phenomena at some locations, or in other words, how well sensors can cover such locations. We will introduce some commonly used sensor coverage models in the next section. Before that, we make an important note here. As abstraction models, sensor coverage models only apply to some types of sensors. There exist some other types of sensors to which coverage considerations may not apply at all.

2.2 Sensor Coverage Models

Sensor coverage models measure the sensing capability and quality by capturing the geometric relation between a space point and sensors. In almost all cases, a sensor coverage model can be formulated as a function of the Euclidean distances (and the angles) between a space point and sensors. The inputs of such a coverage function are the distances (and angles) between a particular space point and sensors' locations, and the output is called *coverage measure* of this space point, which is a nonnegative real number.

We introduce the concept of coverage function in the context of two-dimensional space. Let us consider a space point z and a set of sensors $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$. We use $d(s, z)$ ($d(s, z) \geq 0$) to denote the Euclidean distance between a sensor s and a space point:

$$d(s, z) \doteq \sqrt{(s_x - z_x)^2 + (s_y - z_y)^2} \tag{2.1}$$

in the two-dimensional space, where (s_x, s_y) and (z_x, z_y) are the Cartesian coordinates of the sensor s and the space point z, respectively. We use $\phi(s, z)$ $(0 \leq \phi(s, z) < 2\pi)$ to denote the angle between them. For a sensor s, we draw a horizontal line starting from the sensor and pointing to right. We connect the sensor s and the space point u with another line $\overline{sz}$. Then $\phi(s, z)$ is the anticlockwise angle between the two lines, starting from the horizontal line and ending at the line $\overline{sz}$. Figure 2.1 illustrates an example of d and ϕ. We use $\boldsymbol{d}_n = (d(s_1, z), d(s_2, z), \ldots, d(s_n, z))$ to denote the vector of such distances and $\boldsymbol{\phi}_n = (\phi(s_1, z), \phi(s_2, z), \ldots, \phi(s_n, z))$ to denote the vector of such angles between the set of sensors and the space point. A sensor coverage model can be formulated as a coverage function f mapping $(\boldsymbol{d}_n, \boldsymbol{\phi}_n)$ to a nonnegative real number, that is,

$$f : (\boldsymbol{d}_n, \boldsymbol{\phi}_n) \to \mathbb{R}^+, \tag{2.2}$$

where $\mathbb{R}^+$ stands for the set of nonnegative real numbers. We call $f(\boldsymbol{d}_n, \boldsymbol{\phi}_n)$ the coverage measure of a space point with respect to the sensors $s_1, s_2, \ldots, s_n$. Similar definition can also be applied in three-dimensional space yet with some simple modification of the definition of angles.

Many sensor coverage models have been proposed in the literature. In some models, the inputs of a coverage function are only the distance and angle between a space point and one sensor. In some other models, the inputs of a coverage function are the distances and angles between a space point and more than one sensor. In our view, two types of coverage functions can be classified: One type is a kind of *Boolean* coverage models, where the coverage measure is either 0 or 1 for one space point; and the other can be called *general* coverage models, where the coverage measure can take various nonnegative values. If the angle argument is not included in the coverage function, then such coverage models are called *omnidirectional* coverage models. On the other hand, if it is included, such coverage models are called *directional* coverage models. In what follows, we elaborate some commonly used coverage models in details.

2.2.1 Boolean Sector Coverage Models

The Boolean sector coverage model (sometimes called the sector model), which might be motivated from a directional camera, is a Boolean directional coverage model [12]. Figure 2.2(a) illustrates such a sector model, where ϕ_s is called an *orientational angle*, ω is called a *visual angle* of the sector model, and R_s is called a *sensing range*. The coverage function of the sector model is given by

$$f\big(d(s, z), \phi(s, z)\big) = \begin{cases} 1 & \text{if } d(s, z) \leq R_s \text{ and } \phi_s \leq \phi(s, z) \leq \phi_s + \omega, \\ 0 & \text{otherwise,} \end{cases} \tag{2.3}$$

where $d(s, z)$ is the Euclidean distance between a sensor s and a space point z, and $\phi(s, z)$ is their angle. This coverage function defines a sector: All space points

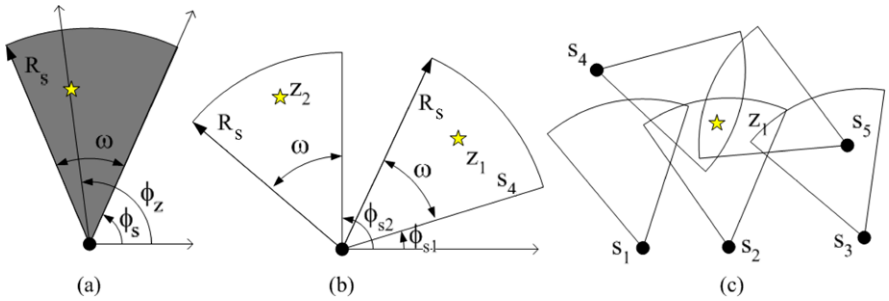

Fig. 2.2 Illustration of (**a**) a directional Boolean sector coverage model; (**b**) a directional Boolean sector coverage model with adjustable orientational angle; and (**c**) a space point being 3-covered by three sectors

within such a sector have the coverage measure of 1 and are said to be *covered* by this sensor. All space points outside such a sector have the coverage measure of 0 and are said to be not covered by this sensor. In Fig. 2.2(a), the space point marked by a star has a coverage measure of 1 and is covered by the sensor.

The orientational angle of a directional sensor might be adjustable after a sensor has been deployed [2, 3, 6]. Obviously, the area that can be covered by such a sensor will be different when it takes different orientational angle. For example, in Fig. 2.2(b), if the sensor takes ϕ_{s1} as its orientational angle, then the space point z_1 is covered, and z_2 is not. If it takes ϕ_{s2} as its orientational angle, then the space point z_1 is not covered, and z_2 is covered.

A space point may be covered by more than one sector [10]. With the Boolean sector coverage model, the coverage measure of a space point relative to a set of sensors can be the addition of the coverage measure of the point relative to each individual sensor. Formally, the coverage function can be defined as

$$f(\boldsymbol{d}_n, \boldsymbol{\phi}_n) = \sum_{i=1}^{n} f_i\big(d(s_i, z), \phi(s_i, z)\big), \tag{2.4}$$

where f_i is the coverage function of a sensor s_i and is given by (2.3). If $f(\boldsymbol{d}_n, \boldsymbol{\phi}_n) = k$ ($k \geq 1$), then we say that the point is *k-covered*. Obviously, if a point is k-covered, it is also $(k-1)$-covered. Figure 2.2(c) illustrates an example of space point being 3-covered, where the space point marked by the star is within the sensing sectors of sensors s_2, s_4 and s_5.

2.2.2 Boolean Disk Coverage Models

The Boolean disk coverage model (often simplified as the disk model) might be the most widely used sensor coverage model in the literature. The coverage function of

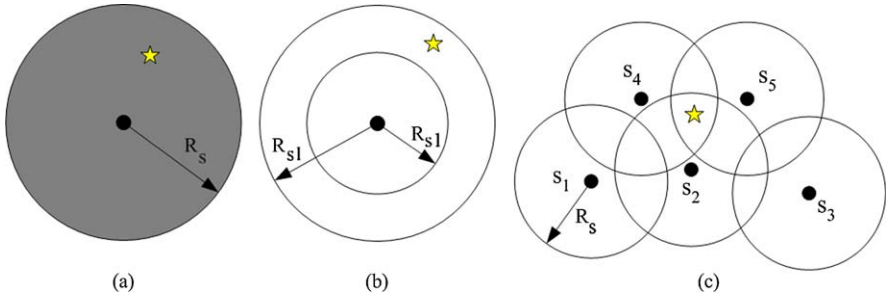

Fig. 2.3 Illustration of (**a**) an omnidirectional Boolean disk coverage model; (**b**) an omnidirectional Boolean disk coverage model with variable sensing ranges; and (**c**) a space point being 3-covered by three disks

the disk model is given by

$$f\big(d(s,z)\big) = \begin{cases} 1 & \text{if } d(s,z) \le R_s, \\ 0 & \text{otherwise,} \end{cases} \tag{2.5}$$

where $d(s,z)$ is the Euclidean distance between a sensor s and a space point z, and the constant $R_s > 0$ is called *sensing range*. Indeed, this function defines a disk (often called a *sensing disk*) centered at the sensor with the radius of the sensing range. Figure 2.3(a) illustrates a disk coverage model. The disk coverage model is an omnidirectional coverage model. All space points within such a disk have the coverage measure of 1 and are said *covered* by this sensor. All space points outside such a disk have the coverage measure of 0 and are said not covered by this sensor.

The sensing range R_s is used to characterize the sensing capability of a sensor. Normally, different sensor types are assumed to have different sensing ranges. Some researchers even argue that a single sensor unit may have different sensing ranges and can choose one sensing range as its working sensing range [4, 22, 33]. For example, Fig. 2.3(b) illustrates a sensor with two sensing ranges, R_{s1} and R_{s2}. The space point marked by the star is not covered if the sensor uses R_{s1} as sensing range; it is covered if the sensor uses R_{s2} as sensing range. It is generally assumed that a sensor consumes more energy when it uses a larger sensing range.

A space point may be located within more than one sensing disks. Under the disk coverage model, the coverage measure of a space point relative to a set of sensors can be the addition of the coverage measure of the point relative to each individual sensor. Formally, the coverage function can be defined as

$$f(\boldsymbol{d}_n) = \sum_{i=1}^{n} f_i\big(d(s_i,z)\big), \tag{2.6}$$

where $f_i(\cdot)$ is the coverage function of a sensor s_i and is given by (2.5). If $f(\boldsymbol{d}_n) = k$, then we say that the point is *k-covered*. Obviously, if a point is k-covered, it is also $(k-1)$-covered. Figure 2.3(c) illustrates an example of space

point being 3-covered, where the space point marked by the star is within the sensing disks of sensors s_2, s_4, and s_5.

2.2.3 Attenuated Disk Coverage Models

Some researchers argue that the sensing quality of a sensor reduces with the increase of the distance away from the sensor [13, 19]. An attenuated disk coverage model is used to capture such attenuated sensing qualities. An example of an attenuated disk coverage model is given by

$$f\big(d(s,z)\big) = \frac{C}{d^{\alpha}(s,z)}, \tag{2.7}$$

where α is the path attenuation exponent, and C a constant. Since it is a nonnegative function, a single sensor enforces its coverage measure to any point in the space. Figure 2.4(a) illustrates such an attenuated disk coverage model. The coverage measure of z_1 is larger than that of z_2, as it is closer to the sensor.

There may be more than one sensor in a sensor field. Under the attenuated disk coverage model, the coverage measure of a space point relative to a set of sensors is the addition of the coverage measure of the point relative to each individual sensor. Formally, the coverage function is modified as

$$f(\boldsymbol{d}_n) = \sum_{i=1}^{n} \frac{C}{d^{\alpha}(s_i,z)}. \tag{2.8}$$

In some cases, only the sensors close to a space point are included in the computation of the above equation for simplification.

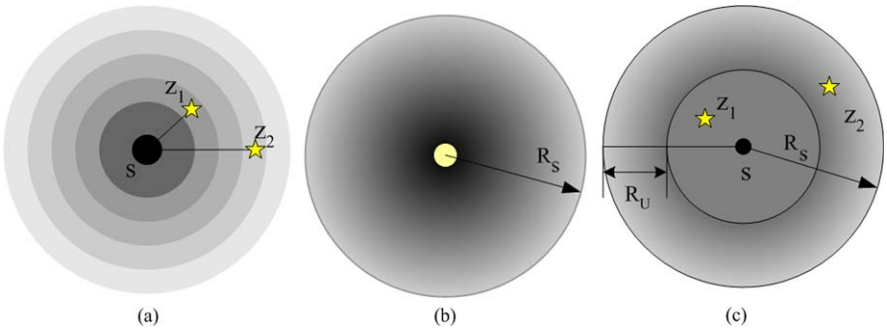

Fig. 2.4 Illustration of (**a**) an attenuated disk coverage model; (**b**) a truncated attenuated disk coverage model; (**c**) a truncated multilevel attenuated disk coverage model

2.2.4 Truncated Attenuated Disk Models

In the attenuated disk coverage model, the coverage measure becomes very small when the distance between a space point and a sensor becomes very large. In such cases, the coverage measure might be neglected, and some approximations can be made by truncating the coverage measure for larger values of distance. For example, Zou and Chakrabarty [32] propose the following truncated attenuated coverage function:

$$f\big(d(s,z)\big) = \begin{cases} Ce^{-\alpha d(s,z)} & \text{if } d(s,z) \le R_s, \\ 0 & \text{otherwise,} \end{cases} \tag{2.9}$$

where α is a parameter representing the physical characteristics of the sensor unit, and R_s the sensing range. Figure 2.4(b) illustrates such a coverage model.

Another truncated attenuated disk model [31] is defined as follows:

$$f\big(d(s,z)\big) = \begin{cases} 1 & \text{if } d(s,z) \le R_s - R_u, \\ e^{-\alpha(d(s,z)-(R_s-R_u))^\beta} & \text{if } R_s - R_u < d(s,z) \le R_s, \\ 0 & \text{if } R_s < d(s,z), \end{cases} \tag{2.10}$$

where R_s is the sensing range, R_u is called the uncertain range, and α and β are constants. The use of R_u is to capture the reducing but not yet vanishing of the sensing quality when the distance between a sensor and a space point increases. Figure 2.4(c) illustrates such a coverage model.

Figure 2.5 illustrates the relation between the coverage measure and sensor–point distance for the aforementioned disk coverage models.

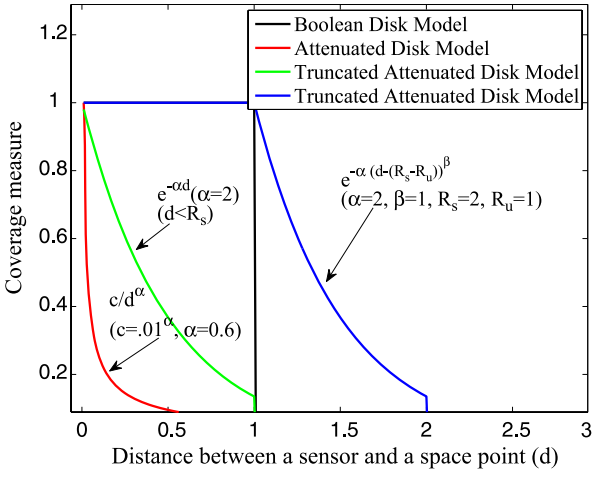

Fig. 2.5 Illustration of the relation between the coverage measure and sensor–point distance for (**a**) Boolean disk coverage model; (**b**) attenuated disk coverage model; and (**c**) and (**d**) truncated attenuated disk coverage model (for the color version, see Color Plates on p. 207)

2.2.5 Detection Coverage Models

An important application of sensor networks is to detect some event occurred at some location. In the context of detection application, the sensing quality of a sensor can be represented by its detection probability. The detection probability of a space point by a single sensor is also related to, among other factors, the distance between them. However, the detection probability of a space point relative to a set of sensors is no longer simply computed as the addition of the detection probability of the point relative to each individual sensor (otherwise, it might be larger than one). Instead, a *value fusion* or *decision fusion* can be used to derive the detection probability. Based on different event scenarios and detection techniques, many detection coverage models have been proposed in the literature [1, 5, 8, 15, 17, 18, 26, 28–30].

Let us consider a general signal propagation model where the signal parameter θ (e.g., the sound pressure of a sound source) attenuates along with the signal propagation. Depending on the hypothesis of whether the target is present (H_1) or not (H_0), the readings at the sensor s_k are given by

$$H_0 : x_k = n_k, \tag{2.11}$$

$$H_1 : x_k = \frac{\theta}{d_k^\alpha} + n_k, \tag{2.12}$$

where α is the attenuation exponent, $d_k^\alpha = d^\alpha(s_k, z)$ is the Euclidean distance between the sensor s_k and the space point z, and n_k is the measurement noise (e.g., circuity thermal noise). It is often assumed that the noise follows a Gaussian distribution with zero mean and variance σ_k^2, denoted by $\mathcal{N}(0, \sigma_k^2)$.

Given the threshold A, a sensor makes its detection decision of whether a target is present by

$$x_k \underset{H_0}{\overset{H_1}{\gtrless}} A. \tag{2.13}$$

That is, if the measurement is larger than A, it decides that a target is present, and if the measurement is less than A, it decides that a target is not present. When a target is present at the space point z, the detection probability P_k^d of the sensor s_k is given by

$$P_k^d = \Pr\left[\frac{\theta}{d_k^\alpha} + n_k \geq A\right] = Q\left(\frac{A - \frac{\theta}{d_k^\alpha}}{\sigma_k}\right), \tag{2.14}$$

where $Q(\cdot)$ is the Q-function defined by

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\, dt. \tag{2.15}$$

Since Q-function is a decreasing function, the detection probability P_k^d decreases when the distance d_k increases. Indeed, (2.14) defines an attenuated disk coverage

model. Furthermore, if we define a threshold for detection probability, P_{th}^d, and only those points with detection probability equal to or larger than such a threshold, i.e., $P_k^d \geq P_{\text{th}}^d$, are considered as covered by this sensor, then we actually define a truncated attenuated disk coverage model. If we do not discriminate the points with detection probability not less than the detection threshold and simply call these points being covered by the sensor, then finally we get a Boolean disk coverage model. In such a case, the points with the detection probability equal to the threshold consist of a circle, and their distances to the sensor are also equal and often regarded as the sensing range R_s. That is,

$$Q\left(\frac{A-\frac{\theta}{R_s^\alpha}}{\sigma_k}\right) = P_{\text{th}}^d \quad \Longrightarrow \quad R_s = \left(\frac{\theta}{A-\sigma_k Q^{-1}(P_{\text{th}}^k)}\right)^{\frac{1}{\alpha}}, \tag{2.16}$$

where $Q^{-1}(\cdot)$ denotes the inverse function of $Q(\cdot)$.

When K sensors are used to cooperatively detect an event, the value fusion technique can be used to compute the detection probability of a space point by these sensors. Let x_k, $k = 1, 2, \ldots, K$, denote the readings of the kth sensor. With the value fusion, we compare the sum of x_k and a threshold to make a decision whether or not a target is present. We assume that all the noises n_k ($k = 1, 2, \ldots, K$) are independent Gaussian noises with zero mean and variance σ^2. When a target is present at the space point z, the detection probability by these sensors is given by

$$P_K^d = \Pr\left[\sum_{k=1}^{K}\left(\frac{\theta}{d_k^\alpha}+n_k\right) \geq \sqrt{K}A\right] = Q\left(\frac{\sqrt{K}A-\sum_{k=1}^{K}\frac{\theta}{d_k^\alpha}}{\sqrt{K}\sigma}\right), \tag{2.17}$$

where $\sqrt{K}A$ is the value fusion threshold. Again, we can use the threshold of detection probability P_{th}^d, and the points with detection probability not less than the detection threshold are called covered by these sensors. In such a case, the covered points by K sensors satisfy the following distance inequality:

$$\sum_{k=1}^{K}\frac{1}{d_k^\alpha} \geq \frac{\sqrt{K}}{R_s^\alpha}, \tag{2.18}$$

where d_k is the distance between a point and a sensor s_k, and R_s given by (2.16). Indeed, (2.18) defines a Boolean detection model for K sensors, that is,

$$f(\boldsymbol{d}_K) = \begin{cases} 1 & \text{if } \sum_{k=1}^{K}\frac{1}{d_k^\alpha} \geq \frac{\sqrt{K}}{R_s^\alpha}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.19}$$

Figure 2.6 marks out the space points that are considered as being covered when using (2.19) ($\alpha = 1.0$) as the coverage model. The points within a disk are considered as being covered when only one sensor is used. They are also considered as being covered when more than one sensor is used. Furthermore, those points colored by yellow (and outside the disks) are not covered by only a single sensor but are considered as being covered by more than one sensor. These additionally covered space

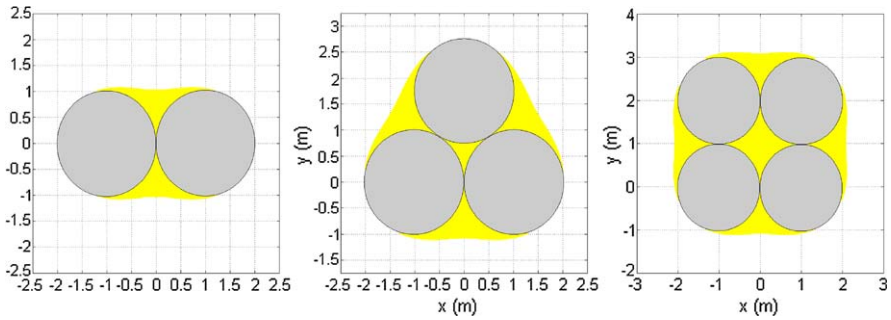

Fig. 2.6 Illustration of the space points covered by using the detection coverage model of (**a**) 2 sensors, (**b**) 3 sensors, and (**c**) 4 sensors (for the color version, see Color Plates on p. 208)

points can be regarded as a kind of cooperation gain by using more than one sensor for a same sensing task.

There are also many decision fusion techniques that can be used to derive the detection probability by a set of sensors. For example, the following decision fusion computes the overall detection probability by a set of sensors:

$$P_K^d = 1 - \prod_{k=1}^{K} \left(1 - P_k^d\right), \tag{2.20}$$

where P_k^d is given by (2.14). Note that P_k^d depends on the distance between a sensor and the space point. Equation (2.20) can also be used to define a coverage model (It has been called as *probabilistic coverage* in some papers [1, 8, 17].) Again, we can set a threshold and define that a point is covered by K sensors $(s_1, \ldots, s_K)$ if its overall detection probability is not less than such a threshold.

Another commonly used decision fusion technique is the *majority voting*. Suppose that there are K sensors, each independently making a local binary decision δ_k. If $x_k \geq A$, then a sensor decides that a target is present, and $\delta_k = 1$; otherwise, a sensor decides that a target is present, and $\delta_k = 0$. Note that δ_k is dependent on the distance between a sensor and a space point. The consensus decision rule is given by

$$\sum_{k=1}^{K} \delta_k \underset{H_0}{\overset{H_1}{\gtrless}} \left\lceil \frac{K}{2} \right\rceil. \tag{2.21}$$

That is, if more than a half of sensors decide a target being present, then the overall decision fusion result is that a target is present; otherwise, the final result is that a target is not present. The consensus detection probability hence is given by

$$P_K^d = \Pr\left[\sum_{k=1}^{K} \delta_k^d \geq \left\lceil \frac{K}{2} \right\rceil\right] = \sum_{j=\lceil \frac{K}{2} \rceil}^{K} \sum_{\text{permutation}} \prod_{k=1}^{j} P_k^d \prod_{k=j+1}^{K} \left(1 - P_k^d\right), \tag{2.22}$$

where P_k^d is given by (2.14). The second sum term is to add up the product of detection probabilities and missing probabilities over all possible permutations over k. Equation (2.22) can also be used to define a coverage model. Also, we can set a threshold and define that a point is covered by K sensors $(s_1, \ldots, s_K)$ if its overall detection probability is not less than such a threshold.

2.2.6 Estimation Coverage Models

Another important application of sensor networks is estimating signal parameters. In the context of estimation application, the sensing quality of a sensor can be represented by its estimation errors. The estimation error of a space point by a single sensor is also related to, among other factors, the distance between them. However, when multiple sensors are used in estimation, the estimation error of a parameter of some signal at a space point is no longer simply computed as the addition of the estimation error of the point relative to each individual sensor. Instead, different estimation techniques can be used, and their estimation errors are also different. Based on different event scenarios and estimation techniques, some estimation coverage models have been proposed [11, 20, 21, 23, 25].

We use a simple signal estimation scenario to illustrate an estimation coverage model. We assume that a signal occurs at some space point z and that its signal parameter θ attenuates along with the signal propagation. For example, θ can be the acoustic amplitude due to a motor engine or due to a leakage of gas barrel. For magnetic wave such as acoustic wave, its amplitude is attenuated when propagating. The measurement of the signal parameter by a sensor s_k is given by

$$x_k = \frac{\theta}{d_k^\alpha} + n_k, \tag{2.23}$$

where α is the attenuation exponent, $d_k^\alpha = d^\alpha(s_k, z)$ is the Euclidean distance between the sensor s_k and the space point z, and n_k is the measurement noise (e.g., circuity thermal noise). It is often assumed that the noise follows a Gaussian distribution with zero mean and variance σ_k^2, denoted by $\mathcal{N}(0, \sigma_k^2)$. We note that this measurement model is the same as the one in (2.12) when the target is present.

A parameter estimator can be used to estimate θ based on the measurements x_k, $k = 1, 2, \ldots, K$. Let $\hat{\theta}$ and $\tilde{\theta} = \hat{\theta} - \theta$ denote the estimate and the estimation error, respectively. If the estimation error is small, the estimate of the signal parameter is obtained with high confidence level. We can use the probability that the absolute value of the estimation is less than or equal to a predefined constant A, i.e., $\Pr[|\tilde{\theta}_K| \leq A]$, to measure how well a point is monitored ($\Pr[|\tilde{\theta}_K| \leq A]$ is called the *information exposure* in [24]). Some standard estimators can be used to perform the estimation. For example, if the *best linear unbiased estimator* (BLUE) estimator [14] is used, the estimate $\hat{\theta}_K$ is given by

$$\hat{\theta}_K = \frac{\sum_{k=1}^K d_k^{-\alpha} \sigma_k^{-2} x_k}{\sum_{k=1}^K d_k^{-2\alpha} \sigma_k^{-2}}, \tag{2.24}$$

and the estimation error $\tilde{\theta}_K$ is given by

$$\tilde{\theta}_K = \frac{\sum_{k=1}^{K} d_k^{-\alpha} \sigma_k^{-2} n_k}{\sum_{k=1}^{K} d_k^{-2\alpha} \sigma_k^{-2}}. \tag{2.25}$$

If we further assume that all noises have the same variances, i.e., $\sigma_k^2 = \sigma^2$ for all $k = 1, 2, \ldots, K$, then we have

$$\Pr\big[|\tilde{\theta}_K| \le A\big] = 1 - 2Q\left(\frac{A}{\sigma}\left(\sum_{k=1}^{K} d_k^{-2\alpha}\right)^{\frac{1}{2}}\right), \tag{2.26}$$

where Q-function is defined in (2.15).

We can see that (2.26) dependence on the distances between the K sensors and the space point and can be used as a coverage function to define an estimation coverage model. Now let us consider that only one sensor is used in estimation. In such a case, (2.26) is given by

$$\Pr\big[|\tilde{\theta}_k| \le A\big] = 1 - 2Q\left(\frac{A}{\sigma} d_k^{-\alpha}\right). \tag{2.27}$$

Since $Q(\cdot)$ is a decreasing function, $\Pr[|\tilde{\theta}_k| \le A]$ decreases as the distance d_k increases. Indeed, (2.27) defines an attenuated disk coverage model. Furthermore, if we define a threshold ϵ ($0 \le \epsilon \le 1$) and if only the points with $\Pr[|\tilde{\theta}| \le A]$ equal to or larger than such a threshold, i.e., $\Pr[|\tilde{\theta}_k| \le A] \ge \epsilon$, are considered as covered by this sensor, then we actually define a truncated attenuated disk coverage model. If we do not discriminate the points within such a disk, then finally we get a Boolean disk coverage model. In such a case, the points with $\Pr[|\tilde{\theta}_k| \le A] = \epsilon$ consist of a circle, and their distances to the sensor are also equal and can be regarded as the sensing range R_s. That is,

$$1 - 2Q\left(\frac{A}{\sigma R_s^{\alpha}}\right) = \epsilon \quad \Longrightarrow \quad R_s = \left(\frac{A}{\sigma Q^{-1}(\frac{1-\epsilon}{2})}\right)^{\frac{1}{\alpha}}, \tag{2.28}$$

where $Q^{-1}(\cdot)$ denotes the inverse function of $Q(\cdot)$.

We can also define a Boolean estimation coverage model of K sensors by comparing $\Pr[|\tilde{\theta}_K| \le A]$ with the threshold ϵ. In such a case, the covered points by K sensors satisfy the following distance inequality:

$$\sum_{k=1}^{K} \frac{1}{d_k^{2\alpha}} \ge \frac{1}{R_s^{2\alpha}}, \tag{2.29}$$

where d_k is the distance between a point and a sensor s_k, and R_s given by (2.28). Indeed, (2.29) defines a Boolean estimation coverage model for K sensors (which

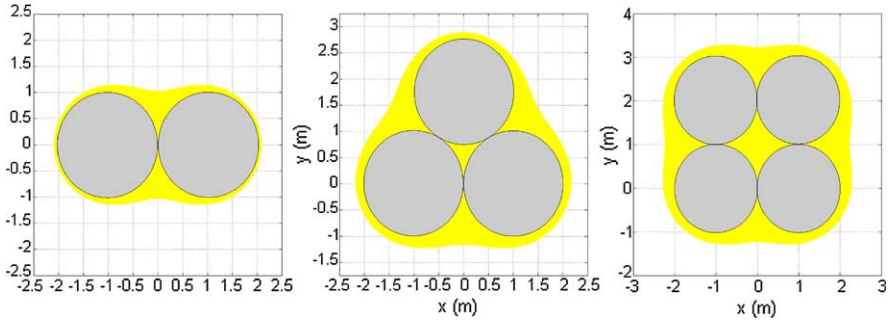

Fig. 2.7 Illustration of the space points covered by using the estimation coverage model of (**a**) 2 sensors, (**b**) 3 sensors, and (**c**) 4 sensors (for the color version, see Color Plates on p. 208)

is called the *information coverage* model in [23]), that is,

$$f(\boldsymbol{d}_K) = \begin{cases} 1 & \text{if } \sum_{k=1}^{K} \frac{1}{d_k^{2\alpha}} \geq \frac{1}{R_s^{2\alpha}}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.30}$$

Figure 2.7 marks out the space points that are considered as being covered when using (2.30) ($\alpha = 1.0$) as the coverage model. It is also seen that when using more than one sensor for the same sensing task (estimation in this case), the covered space points are more than those by only using one single sensor. Again, the increased coverage area can be seen as a kind of cooperation gain.

References

1. Ahmed, N., Kanhere, S.S., Jha, S.: Probabilistic coverage in wireless sensor networks. In: IEEE Conference on Local Computer Networks (LCN), pp. 672–681 (2005)
2. Ai, J., Abouzeid, A.A.: Coverage by directional sensors in randomly deployed wireless sensor networks. Journal of Combinatorial Optimization **11**(1), 21–41 (2006)
3. Cai, Y., Lou, W., Li, M., Li, X.Y.: Target-oriented scheduling in directional sensor networks. In: IEEE Infocom, pp. 1–9 (2007)
4. Cardei, M., Wu, J., Lu, M., Pervaiz, M.O.: Maximum network lifetime in wireless sensor networks with adjustable sensing ranges. In: IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 438–445 (2005)
5. Clouqueur, T., Phipatanasuphorn, V., Ramanathan, P., Saluja, K.K.: Sensor deployment strategy for detection of targets traversing a region. Mobile Networks and Applications **8**(4), 453–461 (2003)
6. Fusco, G., Gupta, H.: Selection and orientation of directional sensors for coverage maximization. In: IEEE Communications Society 6th Annual Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), pp. 1–9 (2009)
7. Hansen, C.H.: Fundamentals of acoustics. In: Goelzer, B., H. Hansen, C.H., Sehrndt, G.A. (eds.) Occupational Exposure to Noise: Evaluation, Prevention and Control. World Health Organization, Geneva (2001)
8. Hefeeda, M., Ahmadi, H.: A probabilistic coverage protocol for wireless sensor networks. In: IEEE International Conference on Network Protocols (ICNP), pp. 1–10 (2007)

9. Kinsler, L.E., Frey, A.R., Coppens, A.B., Sanders, J.V.: Fundamentals of Acoustics. Wiley, New York (2000)
10. Liu, L., Ma, H., Zhang, X.: On directional k-coverage analysis of randomly deployed camera sensor networks. In: IEEE International Conference on Communications (ICC), pp. 2707–2711 (2008)
11. Liu, L., Zhang, X., Ma, H.: Localization-oriented coverage based on Bayesian estimation in camera sensor networks. In: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1–8 (2008)
12. Ma, H., Liu, Y.: Some problems of directional sensor networks. International Journal of Sensor Networks (InderScience) **2**(1–2), 44–52 (2007)
13. Megerian, S., Koushanfar, F.: Exposure in wireless sensor networks: Theory and practical solutions. Wireless Networks **8**, 443–454 (2002)
14. Mendel, J.M.: Lessons in Estimation Theory for Signal Processing, Communications and Control. Prentice Hall, New York (1995)
15. Onur, E., Ersoy, C., Delic, H.: Sensing coverage and breach paths in surveillance wireless sensor networks. In: Phoha, S., La Porta, T.F., Griffin, C. (eds.) IEEE Monograph Sensor Network Operations. IEEE Press, New York (2004) (Chapter 12)
16. Rappaport, T.S.: Wireless Communications: Principles and Practice, 2nd Edition. Prentice Hall, New York (2001)
17. Tian, Y., Zhang, S.F., Wang, Y.: A distributed protocol for ensuring both probabilistic coverage and connectivity of high density wireless sensor networks. In: IEEE Wireless Communications and Networking Conference (WCNC), pp. 2069–2074 (2008)
18. Tsai, Y.R.: Sensing coverage for randomly distributed wireless sensor networks in shadowed environments. IEEE Transactions on Vehicular Technology **57**(1), 556–564 (2008)
19. Veltri, G., Huang, Q., Qu, G., Potkonjak, M.: Minimal and maximal exposure path algorithms for wireless embedded sensor networks. In: ACM International Conference on Embedded Networked Sensor Systems (SenSys), pp. 40–50 (2003)
20. Venkataraman, J., Haenggi, M., Collins, O.: Short noise models for the dual problems of cooperative coverage and outage in random networks. In: The 44th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 1–10 (2006)
21. Wang, R., Cao, W.: Universal information coverage for bandwidth-constrained sensor networks. In: IEEE International Conference on Robotics and Biometrics (ROBIO), pp. 904–907 (2007)
22. Wang, J., Medidi, S.: Energy efficient coverage with variable sensing radii in wireless sensor networks. In: IEEE the 3rd International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 1–8 (2007)
23. Wang, B., Wang, W., Srinivasan, V., Chua, K.C.: Information coverage for wireless sensor networks. IEEE Communications Letters **9**(11), 967–969 (2005)
24. Wang, B., Chua, K.C., Wang, W., Srinivasan, V.: Worst and best information exposure paths in wireless sensor networks. In: International Conference on Mobile Ad-hoc and Sensor Networks (MSN05), also in LNCS, vol. 3794, pp. 52–62 (2005)
25. Wang, B., Chua, K.C., Srinivasan, V., Wang, W.: Information coverage in randomly deployed wireless sensor networks. IEEE Transactions on Wireless Communications **6**(8), 2994–3004 (2007)
26. Wang, W., Srinivasan, V., Chua, K.C., Wang, B.: Energy-efficient coverage for target detection in wireless sensor networks. In: The 6th International Conference on Information Processing in Sensor Networks (IPSN), pp. 313–322 (2007)
27. Wilson, J.S.: Sensor Technology Handbook. Elsevier, Amsterdam (2005)
28. Xing, G., Lu, C., Pless, R., O'Sullivan, J.A.: Co-grid: An efficient coverage maintenance protocol for distributed sensor networks. In: The Third International Symposium on Information Processing in Sensor Networks (IPSN), pp. 414–423 (2004)
29. Yang, G., Qiao, D.: Barrier information coverage with wireless sensors. In: IEEE Infocom, pp. 918–926 (2009)

30. Yang, G., Shukla, V., Qiao, D.: Analytical study of collaborative information coverage for object detection in sensor networks. In: IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON), pp. 144–152 (2008)
31. Zou, Y., Chakrabarty, K.: Sensor deployment and target localization in distributed sensor networks. ACM Transactions on Embedded Computing Systems **3**(1), 61–91 (2004)
32. Zou, Y., Chakrabarty, K.: A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks. IEEE Transactions on Computers **54**(8), 978–991 (2005)
33. Zhou, Z., Das, S.R., Gupta, H.: Variable radii connected sensor cover in sensor networks. ACM Transactions on Sensor Network (ToSN) **5**(1), 1–36 (2009)

Chapter 3
Network Coverage Control

Abstract Sensor coverage model can be considered as a measure of the sensing capability and quality of individual sensors. Network coverage, on the other hand, can be regarded as a collective measure of the quality of service provided by a network of sensor nodes at different geographical locations. A sensor that is performing the sensing task and is covering some space points consumes energy to generate sensing data. In order to reduce data volume and prolong network lifetime, it is much necessary to control which sensors to sense and for how long. Network coverage control, as one of the *middleware services* in the protocol stack, serves such a functionality. According to different application scenarios, network assumptions and operation objectives, the problem of network coverage control can be formulated and solved in many different approaches. The remaining chapters are devoted to elaborating some basic network coverage control problems and their variants. At first, this chapter discusses its motivations and challenges, and also provides a taxonomy of the coverage control problems in sensor networks.

3.1 Motivations and Objectives

In our context of sensor networks, we use *network coverage* to refer to the coverage relation between field-wide points and network-wide sensors. Sometimes, we may regard network coverage as a collective measure of the quality of service provided by a network of sensor nodes at different geographical locations. The sensor unit of a sensor node, which is to perform the sensing task and to produce sensing data, is controllable to be active or inactive (sleep). An active sensor node consumes energy to generate sensing data, and an inactive (sleep) sensor node does not generate any sensing data. We use *network coverage control* to refer to the network-wide control of individual nodes' sensor unit.

The fundamental motivation of network coverage control, which is also its ultimate objective, can be boiled down to the *energy efficiency*. A sensor node, which is normally designed for some specific sensing task (e.g., sensing temperature, acoustic, or seismic signals), is in general with small size, low weight, and limited (and

B. Wang, *Coverage Control in Sensor Networks,*
Computer Communications and Networks,
DOI 10.1007/978-1-84996-059-5_3, © Springer-Verlag London Limited 2010

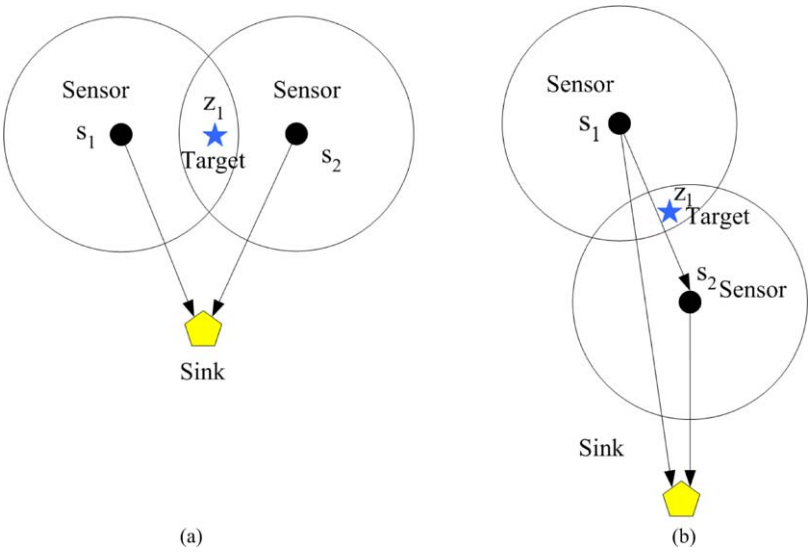

Fig. 3.1 Illustration of benefits from network coverage control in (**a**) a single-hop network; and (**b**) a multi-hop network

nonrechargeable) battery energy. For example, the temperature and relative humidity sensor MEP510 by Crossbow Technology Inc. is of size (cm) $6.35 \times 4.13 \times 3.81$ and weight (grams) 89.8 including a battery [8]. The expected lifetime of a MEP510 is 6 months when default sampling rate is used. A *dead* sensor node that runs out of its energy supply cannot perform sensing task any more. A network consisting of sensor nodes with limited and nonrechargeable power supplies cannot perform the required sensing task if all or a fraction of sensor nodes have used up their energy. Therefore, in order to conserve energy and prolong network lifetime, we need to control which sensor to be active (to cover some space points) and for how long.

A sensor which is performing some sensing task will produce its sensing data at some sampling rate. The sensing data may be first processed locally within the sensor node, and only the results are sent back to the sink. Sometimes, all the sensing data may need to be transmitted back to the sink. The straightforward benefit of coverage control is to reduce the volume of sensing data while still guaranteeing the quality of the sensing task. Other consequent benefits from the reduced network traffic also include the reduced power consumption for data transmission, the reduced transmission collisions, and the reduced data delivery delay.

We use an example to illustrate the network coverage control and its benefits. Figure 3.1 shows two sensor networks used to cover one target and transmit the sensing data back to the sink. Suppose that each sensor node has one energy unit, which can be used to produce two units of data in two time units. In Fig. 3.1(a), we assume that transmitting one unit of data to the sink consumes 0.5 unit of energy. Then sensor node s_1 (s_2) has lifetime of one time unit: It consumes one energy unit to produce one unit data in one unit and send it back to the sink. If we use s_1 and

s_2 to cover the target simultaneously, then the network traffic is two units of data in one unit time, and the total network lifetime is only one time unit. On the other hand, if the quality of service is not affected by allowing only one sensor to monitor the target, then we can let sensor node s_1 to operate for one time unit and sensor node s_2 to operate for another one time unit. With such a control of target coverage, the network traffic is only one unit of data in one unit time, and the network lifetime can be extended to two time units.

Sometimes, network coverage control also incorporates with the routes selection. In Fig. 3.1(b), the sensor node s_1 can use single-hop transmission to send its data to the sink directly or use multi-hop transmission by using sensor node s_2 as its relay to transmit its data to the sink. When the sensor s_1 is used to cover the target, we also need to decide which route it should use to transmit its data. The energy consumption of the individual sensors and the whole network heavily depends on the route selection. Furthermore, if we use sensor s_1 and s_2 sequentially to perform sensing task, we also need to decide how long that sensor is used to cover the target and how much sensing data it can produce.

Coverage control not only can be applied to those already deployed networks for prolonging network lifetime but also can be used before network deployment. The study of the network coverage control can help reducing network setup costs. For small-scale networks where sensor nodes can be manually placed at the desired locations, network coverage control often refers to deciding where are the optimal locations to place the sensor nodes, such that the quality of the sensing task can be guaranteed and the network cost can be minimized. For large-scale networks where sensor nodes are normally randomly scattered within the sensor field, network coverage control usually refers to determining the minimum number (cost) of sensor nodes to provide the required coverage requirements.

As a short summary, the motivations and objectives for network coverage control can be summarized as to reduce network setup costs, conserve node energy consumption, and prolong network lifetime while guaranteeing the specified coverage requirements.

3.1.1 Notes and Comments

Coverage problem can find its origins in computational geometry. For example, the famous *art gallery problem* (e.g., see [12]) is to decide how many cameras are needed and where these cameras should be placed, so that every area of an art gallery can be monitored by at least one camera. Some other famous coverage problems include the minimal disk covering problem [10] (use the minimal number of identical disks to completely cover a disk/rectangle), the line and circle covering problem [6] (the distribution of covered and uncovered segments by a number of randomly distributed disks), the set covering problem [4] (choose a minimal number of subsets to cover all elements), etc. Some of these coverage problems can also find their variants in sensor networks and will be discussed in this book with new problem scenarios, assumptions, and objectives.

3.2 Coverage Control in the Protocol Architecture

Network coverage control is achieved by controlling individual nodes' sensor unit in a network-wide way. In order to implement network coverage control, it is necessary to equip each single sensor node with a *coverage control module*, which is a software implementation of coverage control algorithms to instruct the activation and deactivation of its sensor unit. This section introduces the protocol architecture in a single sensor node and discusses where this coverage control module should be located in this protocol architecture. As there is still not a clearly agreed answer in the literature about this issue, in what follows, we describe only our view, which should be regarded as just one of the many possible solutions.

In our view, we consider the coverage control as one of the middleware modules that sit above the communication protocol stack and below the application stack.

We use Fig. 3.2 to illustrate our view of the software architecture in a sensor node. We note that all these software should have interfaces to the sensor node operating system (OS) and interfaces to hardware such as storage, memory, and central processing unit (CPU). In our view, this architecture consists of three protocol stacks, namely, the *communication stack*, the *middleware stack*, and the *application stack*. Each of these stacks can be further divided into layers or modules. With the term "layer", we emphasize a hierarchy among the layers. With the term "module", we do not impose on a hierarchy structure among the modules. Many of these protocol

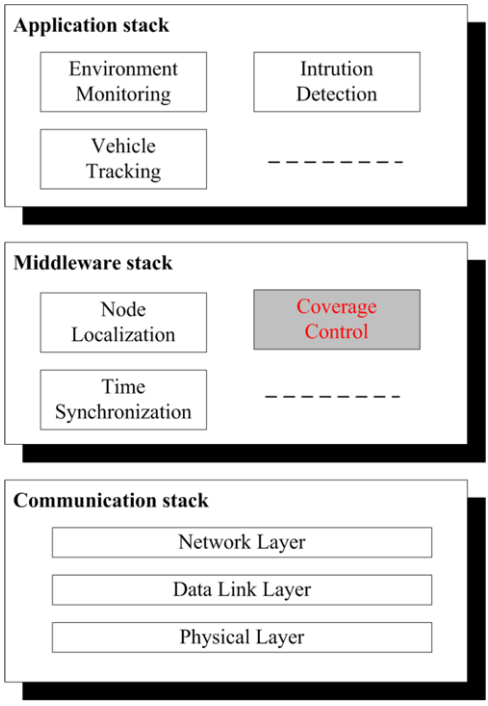

Fig. 3.2 A protocol architecture for sensor nodes. Coverage control is one of the middleware services in between the communication block and application block

layers or modules have been widely discussed in the literature (see, for example, [1, 2, 9]). In what follows, we present only a brief introduction about the functionalities of each layer or module.

The communication stack consists of three protocol layers. From bottom to top, they are *physical layer*, *data link layer*, and *network layer*.

- The physical layer provides mechanical, electrical, functional, and procedural means to establish, maintain, and release physical connections. The connections can be wired or wireless.
- The data link layer can be further divided into two sublayers. The *media access control* (MAC) sublayer sits on top of the physical layer, and the *logical link control* (LLC) sublayer sits on top of the MAC sublayer. The MAC is responsible for controlling the access of individual nodes to the common communication channels, and the LLC is to provide reliable data services to upper layer via error control and retransmissions.
- The network layer is responsible for finding routes from a source node to a destination node, normally the sink. The routing function is especially important in a multi-hop network where a source may need multi-hop transmission to reach the destination node.

The communication stack shown in Fig. 3.2 contains only the lower three layers of the *open system interconnection* (OSI) model [16]. The three layers are generally obligatory to the communication stack. Some researchers have also considered to include the transport layer for wireless sensor networks in order to interconnect with other networks [1].

The application stack consists of many optional modules each corresponding to a particular application of the deployed sensor network. There may have many application modules, and we just list a few example modules in Fig. 3.2. The environment monitoring application is to monitor the sensor field attributes, e.g., temperature, humid, noise, etc. The intrusion detection application is to detect unwanted intrusion to the sensor field, such as private properties or national boundaries. The vehicle tracking application is to use sensor nodes to track the trajectory of a vehicle when it is moving within the senor field.

The middleware stack consists of many optional modules, each performing a particular functionality for node resource allocation and management. Also there may be many middleware modules, and we just present a few example modules in Fig. 3.2. The node localization service is to obtain the node location via, e.g., GPS unit. If GPS unit is not available, then some node localization algorithm is implemented to compute node location. The time synchronization module is to synchronize each node local clock to a global clock. The localization and synchronization modules help to tag the data of a sensor with location and time information, which makes the data more meaningful. The coverage control module, on the other hand, is to dictate when and how long for a node to activate its sensor unit and produce sensing data.

The coverage control module provides the sensing unit activation and deactivation service to applications. For example, if a surveillance application only requires

that each area is monitored by one active sensor, then the coverage control module is to schedule the least number of active sensors to satisfy this surveillance requirement.

The coverage control module uses the services provided by the network layer via an interface. For example, after the coverage control module has decided to become active, it may need to inform its neighbors its decision. This can be done by passing a message to the network layer, and the network layer decides where to send such a message. On the other hand, the network layer also passes the related messages that is received from other nodes to the coverage control module. Indeed, message exchange is essential in the implementation of distributed and localized algorithms. Sometimes, we can also consider a cross-layer design by combining the coverage control module and the network layer: We not only schedule the activation and deactivation of nodes' sensor unit but also select the routes for each active sensor node to send its data back to the sink.

The coverage control module may also use the services from other middleware modules. For example, when it is to decide whether its covered area can also be covered by its active neighbors, it may need the location information for the area coverage computation. In a round-robin manner of coverage decision and maintenance, different nodes may need to synchronize their new round of coverage control, which uses the services provided by the time synchronization module.

As a short summary, we partition the protocol architecture into three stacks, and from bottom to top, they are the communication stack, the middleware stack, and the application stack. We also conclude the coverage control module as one of the services provided by the middleware stack.

3.2.1 Notes and Comments

Middleware can find its origins in traditional distributed computer systems; it has been proposed for bridging the gap between the operating system and applications [14]. Middleware hides the details of different operating systems and is intended to provide transparent and standard services to various applications. Some well-known middleware products include *distributed component object model* (DCOM), *common object requesting broker architecture* (CORBA), etc. The middleware design in the traditional distributed computer systems focuses on the data representation, programming abstraction, and support.

Middleware has also been discussed in the context of wireless sensor networks [5, 7, 13, 15]. For example, Yu et al. [15] argue that the functionalities of middleware should provide standard system services to diverse applications, a runtime environment supporting and coordinating multiple applications, and mechanisms to achieve adaptive and efficient utilization of system resources. Coverage control is one of many network-wide resource management mechanisms which use the services from the communication stack and provide services to applications. Hence we view coverage control module as one of the middleware services in the middleware stack.

3.3 Design Issues of Network Coverage Control

The fundamental objective of network coverage control is to conserve energy; however, different coverage control problems can be formulated according to application scenarios, node capabilities, network assumptions, and performance metrics. In this section, we discuss some design issues of network coverage control.

In a sensor network, sensor nodes with different geographical locations collaborate to achieve high application performance. In different applications, the subject to be covered by a sensor network can be the whole sensor field or only some particular discrete targets within the sensor field. Sensor nodes may be deterministically placed or randomly scattered into a sensor field. In a random deployment, the number of scattered sensor nodes is in general larger than the optimum to compensate for the lack of exact positioning. Roughly saying, a sensor node can be in one of the following three states: transmit and receive state, sensing and processing state, and sleep state. Although different types of sensor nodes may have different energy consumption models, the most expensive energy consumption state in general is the transmit and receive state, and the least one is the sleep state. Since replacing node battery is not feasible in many applications, it is desirable to schedule sensors' states to save battery energy. Besides the sensing task, it is often required in many applications that sensor nodes can deliver their sensory data to a sink for further processing. Sensor nodes can also be assumed to have different sensing, processing, and communication capabilities, which also impact on the coverage problems. A stationary sensor node cannot move once it has been deployed. A mobile sensor node, on the other hand, is equipped with locomotive and can move around after been deployed. Using mobile sensor nodes can help to improve network coverage.

In what follows, we classify the following design issues for coverage problems: namely, *coverage type*, *deployment method*, *sensor heterogeneity*, *activity scheduling*, *coverage degree*, *coverage ratio*, *network connectivity*, and *performance metric*.

Coverage Type Coverage type refers to the subject to be covered by a sensor network. Cardei et al. [3] argue that, according to the subject to be covered, coverage in sensor networks can be classified into three types, namely, *point* (*target*) *coverage*, *area coverage*, and *barrier coverage*. Figure 3.1 illustrates an example of point coverage, and Fig. 3.3 illustrates examples for area coverage and barrier coverage. In the point coverage problems, targets are often modeled as a set of discrete points within the sensor field, and it concerns how to cover these targets. Area coverage problem, on the other hand, equally treats every point in the sensor field and addresses the problem of how to efficiently cover the whole sensor field. Barrier coverage is different from point coverage and area coverage in that the subjects to be covered are not known before node deployment. Instead, it concerns with constructing a barrier for intrusion detection or finding a penetration path across the sensor field with some desired property. For example, the maximal breach path problem presented in [11] concerns with the barrier coverage, which tries to find a path such that each point on the path has the maximal Euclidean distance to its closest sensor. In the rest of the book, we will elaborate the coverage control problems in each of the three coverage types.

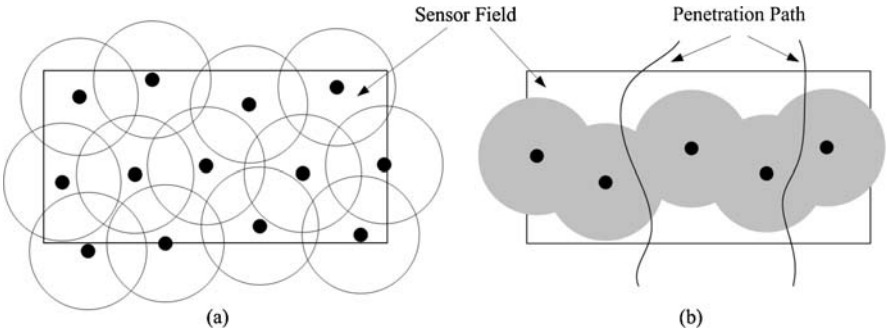

Fig. 3.3 Examples of (**a**) area coverage and (**b**) barrier coverage

Deployment Method Deployment method concerns with how a sensor network is constructed. In general, a sensor network can be constructed by deterministically placing sensor nodes at desired locations or by randomly scattering sensor nodes into the sensor field. In deterministic sensor deployment, the common objective is to place the least number of sensor nodes (the minimum network setup cost) to achieve the application coverage requirement. Deterministic sensor placement can be applied to a small to medium sensor network in a friend environment. When the network size is large or the sensor field is remote and hostile, random sensor deployment might be the only choice. There are two commonly used random deployment models. One is the *random uniform deployment* of N sensor nodes such that each node has equal likelihood of falling at any location in the sensor field, independently of the other nodes. The other is the *random Poisson deployment* with density λ, where the process for deploying sensor nodes is a stationary Poisson point process, and the number of sensors in any subregion is Poisson distributed and mutually independent of each other in different disjoint subregions [6]. In the random deployment, an interesting question is: what is the minimum number of sensor nodes that need to be scattered so that complete coverage can be achieved. This is the *critical sensor density* problem which will be studied in the later chapter.

Node Heterogeneity Node heterogeneity refers to that sensor nodes have different sensing, processing, or communication capabilities. In other words, sensor nodes are heterogeneous. For example, some sensor nodes are resource rich nodes with more power supply or are equipped with better sensing, processing, and communication units. In the context of sensing disk coverage model, some sensors may cover a disk with larger radius than other sensors'. A heterogeneous sensor network may also consist of both stationary or mobile sensor nodes. Stationary sensor nodes will be fixed on its location once deployed, and mobile nodes can move around after the initial deployment. A mobile node is in general more expensive than its stationary compeers and is normally assumed to be resource rich. Many network performance metrics can be greatly improved by using a few of mobile nodes. In the context of network coverage, mobile nodes can be used to heal coverage hole that is not covered by any stationary sensor or to maximize area coverage by relocating mobile sensor nodes. We will study the design of movement strategy in the later chapter.

Activity Scheduling Activity scheduling is to schedule the activation and deactivation of nodes' sensor units. In a randomly deployed sensor network, the scattered sensor nodes may be more than the optimum. If the area covered by one sensor can also be covered by other sensors, such a sensor can be as redundant and can be temporarily transited into the energy saving sleep state. Hence the objective of activity scheduling is to decide which sensors to be in which states and for how long time, so that application coverage requirement can be guaranteed and network lifetime can be prolonged. In the context of coverage problems, *network lifetime* is often defined as the period from the network setup time to the time that the deployed network cannot provide adequate coverage (e.g., the coverage ratio less than a predefined threshold). Many *distributed algorithms* and *centralized algorithms* for activity scheduling have been proposed in the literature, yet based on different assumptions and objectives. In the distributed algorithms, the decision process is localized in each individual sensor node, and only information from neighboring nodes is used for the activity decision. In centralized algorithms, a central controller makes all decisions and distributes the results to sensor nodes. For large sensor networks with dynamic topologies, distributed algorithms are more preferred than centralized ones. We will study the activity scheduling problem and elaborate some example algorithms in the later chapter.

Coverage Degree Coverage degree describes how a point is covered. For example, in the sensing disk coverage model, coverage degree refers to how many sensors cover a point. A point is called k-covered if it is within k distinct sensors' coverage disks. Using more than one sensor to cover a point can improve coverage robustness. If a point is covered by k sensors, then it can tolerate up to $k-1$ failed sensors. A similar definition can also be applied to other coverage models. For the sake of simplicity, in the whole book, by a covered point we mean that this point is covered by at least one sensor. We will explicitly state higher coverage degree when necessary. Coverage degree is considered as one of the application coverage requirements to be observed by coverage control algorithms.

Coverage Ratio Coverage ratio measures how much area of a sensor field or how many targets satisfy the application requirement of coverage degree. For example, if eight out of ten targets are covered, then the coverage ratio is 80%. We sometimes use *complete coverage* to refer to 100% coverage ratio, that is, every point within sensor field (or every target in the target set) achieves the required coverage degree. Similarly, we use *partial coverage* to state the situation that not all points in the sensor field (or not all targets in the target set) can be covered with the required coverage degree. Complete area coverage is a very strict requirement, which normally requires a large amount of sensors to be deployed into the sensor field. Instead of asking for 100% coverage ratio, we may allow that some points or targets are not covered and trade-off the coverage ratio with less active sensors. Coverage ratio is often regarded as one of the application coverage requirements to be observed by coverage control algorithms.

Network Connectivity Network connectivity concerns with how to guarantee that all sensor nodes can find a route to the sink. Although normally this is the task of the network layer, it may also be incorporated into the design of coverage control algorithms as a cross-layer approach. In wireless sensor networks, wireless sensor nodes communicate via their radio transceivers. Two wireless nodes are directly connected if they can transmit and receive the data to and from each other via radio channel. Two nodes can also be connected by multi-hop transmissions with some other nodes serving as *relays*. A connected network ensures that the sensing data of any sensor node can be transmitted to other nodes and the sink, possibly via multi-hop transmissions. A commonly used transmission model is the disk model where a node can communicate with other nodes within a disk centered at itself with the radius of its *communication range*. In such a case, a unit disk graph can be used to describe the network. Some other transmission model also allows variable transmission powers and hence variable transmission ranges, and allows transmission errors. The transceiver unit is in general independent of the sensor unit in a sensor node, and the sensing range and the communication range of a sensor can be with different distances. The design of sensor activity scheduling can be coupled with the network connectivity, and we will study some examples of such cross-layer activity scheduling algorithms in the later chapter. Network connectivity sometimes is also considered as one of the application coverage requirements to be observed by coverage control algorithms.

Performance Metric Performance metrics are used to compare different coverage control algorithms. For example, if two activity scheduling algorithms can achieve the same level of coverage degree and coverage ratio, the one selects a fewer number of nodes is often considered as the better one. Furthermore, with different problem settings, the performance metrics can be different for different coverage problems. The most commonly used performance metric is to use the least number of sensor nodes to achieve application coverage requirements or, on the other hand, to achieve the maximum coverage when giving a fixed number of nodes. Some other performance metrics include the coverage lifetime which is the maximum time to ensure application coverage, the coverage intensity which is defined as the average time ratio between covered and uncovered period for a point, the movement cost which is used to compare movement strategies, and so on. Sometimes, network connectivity, though independently controlled by radio transceiver, is also used as a performance metric for activity scheduling. Some other aspects in general algorithm design such as computation complexity, message overhead, and approximation ratio can also be used for evaluating coverage control algorithms.

3.4 A Taxonomy for Network Coverage Problems

We have discussed the motivations, objectives, and design issues of network coverage control. In this section, we provide our taxonomy for various coverage control problems in sensor networks. Before presenting the taxonomy, we first provide a

definition for coverage control. Although we have used the term "network coverage control", there is no clearly agreed definition in the literature. In what follows, we provide our definition, which should be regarded as one of many other possible statements.

Network coverage control *is the art of identifying network-wide coverage characteristics and coordinating nodes' sensing functionality in a network-wide way to achieve application coverage requirement while reducing node energy consumption*.

Although this definition is quite general, it captures the *network-wide* characteristics for the coverage problems in sensor networks. Furthermore, this definition states two main functionalities for network coverage control. One is to identify the *coverage characteristics* of sensor networks, and the other is to control (to maintain or change) the coverage characteristics for the deployed sensor network. Coverage characteristics of a sensor network include the coverage degree and coverage ratio of the space points, as well as the coverage relation between space points and sensor nodes. Identifying the coverage characteristics for a sensor network helps to improve the network coverage configuration and facilitates the control of the network coverage. Controlling the coverage characteristics for a sensor network can help to reduce energy consumption and extend network coverage lifetime.

Note that our definition does not impose any constraint on the scenarios and mechanisms for network coverage control. Network coverage control can be executed before or after network deployment and can also be implemented by centralized or distributed algorithms.

The rest of the book will introduce various network coverage control problems and their solution approaches. Before that, we provide a taxonomy for the network coverage control problems, which is depicted in Fig. 3.4. Again, we note that this taxonomy is from our view and should be recognized as one of many other possible proposals. This taxonomy also presents the structure of the book. In this book, we classify coverage problems into three categories based on the coverage type, namely, *point coverage problems* (Part II), *area coverage problems* (Part III), and *barrier coverage problems* (Part IV). In each of these problems, we will study the representative coverage problems studied in the literature, including application scenarios, problem formulations, mathematical models, and solution approaches. We also summarize various problem variants in each problem category for interested readers.

Point Coverage Problems Part II is devoted to the point coverage problems. In the point coverage problem, the subject to be covered is a set of discrete points. These points can be used to represent some physical targets in the sensor field.

- Chapter 4 studies the node placement optimization problem for coverage configuration before network deployment, where the objective is to find the optimal locations (among the available locations) to place sensor nodes to minimize network cost (e.g., the number of nodes).
- Chapter 5 investigates the coverage lifetime maximization problem by controlling coverage characteristics in a randomly deployed network, where the objective is to optimally schedule sensors activities in order to extend network lifetime.

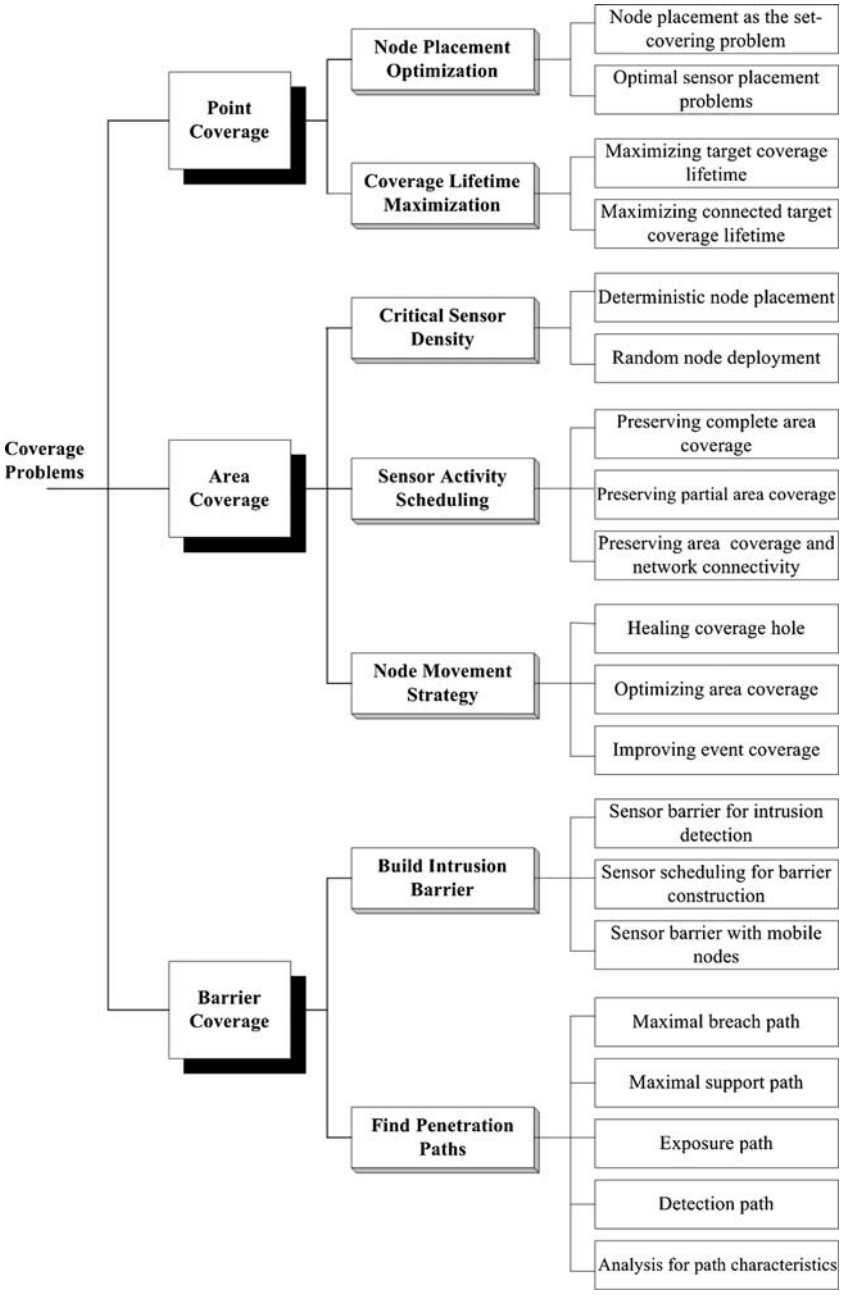

Fig. 3.4 A taxonomy for network coverage control problems in sensor networks

Area Coverage Problems Part III is dedicated to the area coverage problems. In the area coverage problem, the subject to be covered is the whole sensor field. Complete area coverage requires that every space point within the sensor field should be covered, while partial area coverage allows that some space points need not to be covered.

- Chapter 6 discusses the *critical sensor density* (CSD) problem for coverage configuration before network deployment, where the objective is to find the least number of sensor nodes per unit area to provide complete coverage for the whole sensor field. In deterministic node placements, the objectives are to determine not only the least number of nodes to be placed but also their locations. In random node deployments, the objective is to determine the critical sensor density: Whenever the number of sensor nodes to be scattered is not less than this critical density times the area of the sensor field, all points of the sensor field can be covered almost surely in every random deployment.
- Chapter 7 looks into the sensor activity scheduling problem of controlling network coverage characteristics in a randomly deployed network, where the objective to identify coverage redundant sensors and schedule sensors' activity in order to prolong the network lifetime. This problem has been intensively studied in the literature. Based on different assumptions and objectives, many problem variants have been formulated and different algorithms have been proposed.
- Chapter 8 introduces the node movement strategy problem for sensor networks containing mobile nodes, where the objective is to leverage mobile nodes to control network coverage. Mobile nodes change network coverage characteristics via moving to the desired locations. Although network coverage can be greatly improved, the product cost and the moving cost of mobile nodes need to be minimized. The design of node movement strategy should balance between network coverage and movement cost.

Barrier Coverage Problems Part IV discusses the barrier coverage problems. In the barrier coverage problem, the objective is to identify the desired coverage characteristics, if it exists, for a sensor network.

- Chapter 9 examines the coverage problems of building intrusion barriers. A sensor barrier for intrusion detection is similar to an electric fence, which detects an intrusion of a mobile object when it traverses from one side to the other side of the sensor field. The trajectory of an intrusion mobile object is called its traverse path. In barrier coverage, it is not necessary to coverage all space points in the sensor field. Instead, it requires that the covered points can form a barrier, stretching across the sensor field and intersecting with every potential traverse path.
- Chapter 10 reviews the coverage problems of finding penetration paths. A penetration path is a continuous curve with arbitrary shape, spanning from one side to the other side of a sensor field. We assign a coverage measure (a real value) to represent the coverage characteristics of a single space point, e.g., the Euclidean distance between a point and its closest sensor. The objective is to identify such

a penetration path on which every single point satisfies the required coverage measure.

References

1. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: A survey. Computer Networks **39**(4), 393–422 (2002)
2. Callaway, E.H.: Wireless Sensor Networks: Architectures and Protocols. CRC Press, Boca Raton (2004)
3. Cardei, M., Wu, J.: Coverage in wireless sensor networks. In: Ilyas, M., Mahgoub, I. (eds.) Handbook of Sensor Networks. CRC Press, Boca Raton (2004) (Chapter 19)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd Edition. MIT Press, Cambridge (2001)
5. Hadim, S., Mohamed, N.: Middleware challenges and approaches for wireless sensor networks. IEEE Distributed Systems Online **7**(3), 1–23 (2006)
6. Hall, P.: Introduction to the Theory of Coverage Processes. Wiley, New York (1988)
7. Heinzelman, W.B., Murphy, A.L., Carvalho, H.S., Perillo, M.A.: Middleware to support sensor network applications. IEEE Network **18**(1), 15–21 (2004)
8. Inc., C.T.: Mep 510 datasheet. http://www.xbow.com (2005)
9. Karl, H., Willig, A.: Protocols and Architectures for Wireless Sensor Networks. Wiley, New York (2005)
10. Kershner, R.: The number of circles covering a set. American Journal of Mathematics **61**(3), 665–671 (1939)
11. Meguerdichian, S., Koushanfar, F., Potkonjak, M., Srivastava, M.B.: Coverage problems in wireless ad-hoc sensor networks. In: IEEE Infocom, vol. 3, pp. 1380–1387 (2001)
12. O'Rourke, J.: Art Gallery Theorems and Algorithms. Oxford University Press, Oxford (1987)
13. Römer, K., Kasten, O., Mattern, F.: Middleware challenges for wireless sensor networks. ACM SIGCOMM Computer and Communications Review **6**(4), 59–61 (2002)
14. Wikipedia: Middleware. http://en.wikipedia.org/wiki/Middleware (2009)
15. Yu, Y., Krishnamachari, B., Prasanna, V.K.: Issues in designing middleware for wireless sensor networks. IEEE Network **18**(1), 15–21 (2004)
16. Zimmermann, H.: Osi reference model—the iso model of architecture for open systems interconnection. IEEE Transactions on Communications **28**(4), 425–432 (1980)

Part II
Target Coverage Problems

Chapter 4
Node Placement Optimization

Abstract In the point coverage problem, the subject to be covered is a set of discrete space points. These points can be some particular space points to represent the sensor field (e.g., the vertices of a grid) or are used to model some physical targets in the sensor field (e.g., the missile launchers in a battlefield). In order to cover these points, sensor nodes can be deterministically placed or randomly deployed in the sensor field. In a random network deployment, it is possible that some nodes cannot cover even a single target point. On the other hand, if the sensor field is friendly reachable and the network size is not too large, deterministic node placement is to place the nodes only at the desired locations so that each sensor node can cover at least one target. The objective of a deterministic node placement can be summarized as to answer the following question:

Where are the optimal locations (*among the available locations*) *to place sensor nodes so that the number of nodes* (*or the network cost*) *can be minimized and the point coverage requirements can be satisfied?*

The problem of placing the least number of sensors to cover all target points is a variant of the canonical set-covering problem and is discussed in the first part of this chapter. The second part of this chapter introduces some variants of the optimal node placement problem with different coverage requirements.

4.1 Node Placement as the Set-Covering Problem

The locations of the targets (i.e., the space points) to be covered are assumed to be known before placement, and the available locations to place nodes (called *sites*) are limited. The problem of placing the least number of sensors to cover all discrete targets can be equated to the canonic *set-covering problem* [3]. Let X denote the finite set of targets with known locations. A sensor node can only be placed at one of the available sites and can cover at least one target if it is placed at this site. Furthermore, all targets can be covered if all of these available sites are occupied by sensors. Let $\mathcal{F}$ denote a family of subsets of X, and let $S \in \mathcal{F}$ be its element. The cardinality of $\mathcal{F}$ is the number of available sites to place sensors. Every element

B. Wang, *Coverage Control in Sensor Networks,*
Computer Communications and Networks,
DOI 10.1007/978-1-84996-059-5_4, © Springer-Verlag London Limited 2010

of $\mathcal{F}$ corresponds to the set of targets that can be covered if a sensor is placed at one site. We say that S covers some targets. The set-covering problem is to find a minimum-size subset $\mathcal{C} \subseteq \mathcal{F}$ whose elements cover all targets of X, that is,

$$X = \bigcup_{S \in \mathcal{C}} S. \tag{4.1}$$

We say that any $\mathcal{C}$ satisfying (4.1) covers X or that $\mathcal{C}$ is a set-cover of X. The size of $\mathcal{C}$ is defined as the number of sets it contains, rather than the number of individual elements in these sets.

Figure 4.1 illustrates that nine sites (the black squares) are available to place sensor nodes, and 15 targets (the red stars) need to be covered. As seen from the figure, the optimal node placement problem can be simply converted as a set-covering problem: Among the nine available sites, select the minimum number of sites to place sensors so that all targets are covered. The decision version of the set-covering problem (whether a subset of $\mathcal{F}$ with a given size k can cover X) is an NP-complete problem. The following classical greedy algorithm (adapted from [3]) can be used to solve the set-covering problem in time polynomial in $|X|$ and $|\mathcal{F}|$. The greedy algorithm works by selecting, at each step, the set S that covers the largest number of remaining elements that are uncovered.

GREEDY-SET-COVER$(X, \mathcal{F})$ [3]

1. $U \leftarrow X$.
2. $\mathcal{C} \leftarrow \emptyset$
3. **while** $U \neq \emptyset$
4. **do** select an $S \in \mathcal{F}$ that maximizes $|S \cap U|$.
5. $U \leftarrow U - S$
6. $\mathcal{C} \leftarrow \mathcal{C} \cup \{S\}$
7. **return** $\mathcal{C}$

The set U contains, at each stage, the set of remaining uncovered targets. The set $\mathcal{C}$ contains the cover being constructed. Line 4 is the greedy decision-making step. A subset S is chosen if it covers as many uncovered targets as possible (with ties broken arbitrarily). After S is selected, its elements (the already covered targets) are removed from U, and S is placed in $\mathcal{C}$. When the algorithm terminates, the set $\mathcal{C}$ contains a subfamily of $\mathcal{F}$ that covers X.

The number of iterations of the loop on Steps 3–6 is bounded from above by $\min(|X|, |\mathcal{F}|)$, and the loop body can be implemented to run in time $O(|X||\mathcal{F}|)$. Hence the algorithm GREEDY-SET-COVER can be implemented to run in time $O(|X||\mathcal{F}| \min(|X|, |\mathcal{F}|))$. Furthermore, it produces a set-cover with the size not too much larger than an optimal set-cover. Let us denote the Kth harmonic number $H(K) = \sum_{k=1}^{K} \frac{1}{k}$, and we define $H(0) = 0$. The following theorem provides an approximation ratio of the greedy algorithm.

Theorem 1 (Cormen et al. [3], Theorem 35.4) *GREEDY-SET-COVER produces a set-cover with an approximation ratio* $H(\max\{|S| : S \in \mathcal{F}\})$ *to the optimal set-cover.*

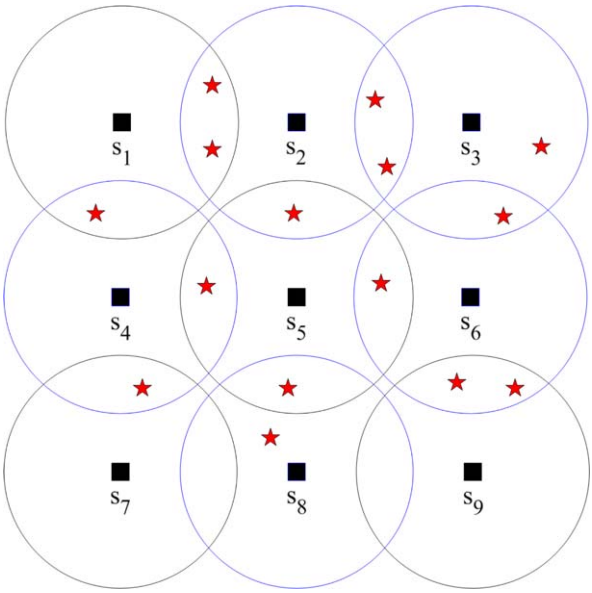

Fig. 4.1 (Color online) An example of the set-covering problem. There are 15 targets (*the red stars*) to be covered. Sensors can put only at nine available sites (*the black squares*), i.e., $s_1, \ldots, s_9$. The greedy algorithm produces a cover of size 5 by selecting the set s_2, s_6, s_4, s_8, and s_3 in sequence

Proof Let $\mathcal{C}^*$ denote the optimal set-cover, and let $\mathcal{C}$ denote the set-cover returned by the algorithm. We need to prove

$$|\mathcal{C}| \leq |\mathcal{C}^*| H\left(\max\left\{|S| : S \in \mathcal{F}\right\}\right). \tag{4.2}$$

We assign a cost of 1 to each set selected by the algorithm and distribute this cost over the targets covered for the first time. Let S_i denote the ith subset selected by GREEDY-SET-COVER; the algorithm incurs a cost of 1 when it adds S_i to $\mathcal{C}$. We spread this cost of selecting S_i evenly among the elements covered for the first time by S_i. Let c_x denote the cost allocated to element $x \in X$. Each element is assigned a cost only once when it is covered for the first time. If x is covered for the first time by S_i, then

$$c_x = \frac{1}{|S_i - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})|}.$$

At each step of the algorithm, 1 unit of cost is assigned, and so

$$|\mathcal{C}| = \sum_{x \in X} c_x.$$

The cost assigned to the optimal set-cover is

$$\sum_{S \in \mathcal{C}^*} \sum_{x \in S} c_x.$$

Since each $x \in X$ is in at least one set $S \in \mathcal{C}^*$, we have

$$|\mathcal{C}| = \sum_{x \in X} c_x \leq \sum_{S \in \mathcal{C}^*} \sum_{x \in S} c_x. \tag{4.3}$$

Consider any set $S \in \mathcal{F}$ and $i = 1, 2, \ldots, |\mathcal{C}|$, and let

$$u_i = \left|S - (S_1 \cup S_2 \cup \cdots \cup S_i)\right|$$

be the number of elements in S remaining uncovered after $S_1, S_2, \ldots, S_i$ have been selected by the algorithm. We define $u_0 = |S|$ to be the number of elements of S that are all initially uncovered. Let k be the least index such that $u_k = 0$, so that each element in S is covered by at least one of the sets $S_1, S_2, \ldots, S_k$. Then $u_{i-1} \geq u_i$, and $u_{i-1} - u_i$ elements of S are covered for the first time by S_i for $i = 1, 2, \ldots, k$. Thus,

$$\sum_{x \in S} c_x = \sum_{i=1}^{k} (u_{i-1} - u_i) \times \frac{1}{|S_i - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})|}.$$

Because the greedy choice of S_i guarantees that S cannot cover more new elements than S_i does (otherwise, S would have been chosen instead of S_i), we have

$$\left|S_i - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})\right| \geq \left|S - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})\right| = u_{i-1}.$$

Consequently, we bound $\sum_{x \in S} c_x$ as follows:

$$\begin{aligned}
\sum_{x \in S} c_x &\leq \sum_{i=1}^{k} (u_{i-1} - u_i) \times \frac{1}{u_{i-1}} \\
&= \sum_{i=1}^{k} \sum_{j=u_i+1}^{u_{i-1}} \frac{1}{u_{i-1}} \\
&\leq \sum_{i=1}^{k} \sum_{j=u_i+1}^{u_{i-1}} \frac{1}{j} \quad \text{(because } j \leq u_{i-1}\text{)} \\
&= \sum_{i=1}^{k} \left(\sum_{j=1}^{u_{i-1}} \frac{1}{j} - \sum_{j=1}^{u_i} \frac{1}{j} \right) \\
&= \sum_{i=1}^{k} \big(H(u_{i-1}) - H(u_i) \big) \\
&= H(u_0) - H(u_k) \quad \text{(because the sum telescopes)} \\
&= H(u_0) - H(0) \\
&= H(u_0) \quad \text{(because } H(0) = 0\text{)} \\
&= H\big(|S|\big).
\end{aligned} \tag{4.4}$$

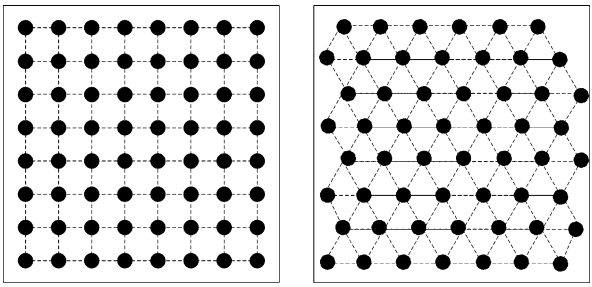

Fig. 4.2 Illustration of grid placement and coverage: (**a**) *a square grid* and (**b**) *a triangular grid*. Nodes can only be placed at the grid vertices, and all grid vertices are required to be covered by the placed nodes

Therefore, by combining (4.3) and (4.4), we have

$$\begin{aligned} |\mathcal{C}| &\leq \sum_{S \in \mathcal{C}^*} \sum_{x \in S} c_x \\ &\leq \sum_{S \in \mathcal{C}^*} H\big(|S|\big) \\ &\leq |\mathcal{C}^*| H\Big(\max\big\{|S| : S \in \mathcal{F}\big\}\Big), \end{aligned}$$

which completes the proof. □

The point coverage problem can be linked to the area coverage problem via a grid approach: If all the vertices (or called *grid points*) of a well-defined grid embedded within the sensor field are covered, then the whole sensor field is said to be completely covered. Figure 4.2 illustrates two grid examples, where sensor nodes are to be placed at those grid points, and all grid points are required to be covered. A grid point is covered by the sensor node located at this grid point. Some regular polygons can be used to tile-up the whole sensor field, and the sensor field is completely covered if sensors are placed at the vertices of these regular polygons. Indeed, the optimal tessellation using regular triangles with side length equal to $\sqrt{3}R_s$ achieves the minimum sensor density for complete area coverage, where R_s is the radius of the sensor sensing disk. The placement problem is trivial if a sensor at a grid point can only cover this grid point. In such a case, every grid point should be occupied by a sensor. On the other hand, for a given grid where a sensor node located at this grid point can also cover a few of its neighboring grid points, the problem of placing the least number of sensors only at the grid points such that the grid points are covered is still an NP-complete problem [9]. As to be discussed in the next section, the node placement problem can be modeled as an optimization problem.

4.2 Optimal Sensor Placement Problems

The problem of placing the least number of sensor nodes to cover all targets is actually an optimization problem. Many optimization techniques can be used to model

and solve the node placement problem. We first present the optimization modeling for the placement problem discussed in the previous section and then introduce some variants of the placement problem with different coverage requirements.

4.2.1 Modeling Node Placement

Suppose there are I sites and J targets. We use subscript i to index a site and j to index a target. Let x_i denote the indicator function of whether a sensor is placed at the site i, i.e.,

$$x_i = \begin{cases} 1 & \text{if a sensor is placed at site } i, \\ 0 & \text{otherwise.} \end{cases}$$

Let δ_{ij} denote an indicator function of whether a target j can be covered by a sensor located at the site i, i.e.,

$$\delta_{ij} = \begin{cases} 1 & \text{if target } j \text{ is covered by a sensor located at site } i, \\ 0 & \text{otherwise.} \end{cases}$$

The problem of placing the least number of sensors to cover all targets can be formulated as the following *integer linear programming* (ILP) problem:

$$\text{Minimize:} \quad \sum_{i=1}^{I} x_i \tag{4.5}$$

$$\text{Subject to:} \quad \sum_{i=1}^{I} \delta_{ij} > 0, \quad j = 1, \ldots, J \tag{4.6}$$

$$x_i \in \{0, 1\}, \quad i = 1, \ldots, I. \tag{4.7}$$

The constraint (4.6) is to guarantee that all targets should be covered by at least one sensor.

A generalization of the above ILP problem is to minimize the network cost if different sensor nodes have different costs and coverage capabilities [1, 2, 14, 20, 22]. Sometimes, we may have other constraints such as the distance between any pair of sensor nodes should not be too close [15]. For example, in the scenario of placing different types of sensors, suppose that we have A types of sensors, each type with cost c_a and coverage distance r_a, $a = 1, \ldots, A$. Normally, a larger coverage range corresponds to a larger cost. Let $D_a(i)$ denote the set of targets that can be covered by a type a sensor placed at the site i, i.e.,

$$D_a(i) = \{j \mid d(i, j) \le r_a\}, \quad i = 1, \ldots, I,$$

where $d(i, j)$ is the Euclidean distance between a site i and a target j. Again, we use $x_i^a = 1$ to denote that a type a sensor been placed at site i and $x_i^a = 0$ otherwise.

Furthermore, the coverage requirement is to cover each target with at least k sensors. The objective is to minimize the total sensor placement cost

$$\text{Minimize:} \quad \sum_{i=1}^{I}\sum_{a=1}^{A} c_a x_i^a,$$

$$\text{Subject to:} \quad \sum_{a=1}^{A}\sum_{j\in D_a(i)} x_i^a \geq k, \quad i = 1,\ldots,I$$

$$\sum_{a=1}^{A} x_i^a \leq 1, \quad i = 1,\ldots,I.$$

The first constraint ensures that each target is covered by at least k sensors, and the second constraint is to ensure that each site can be occupied by at most one sensor.

It is also possible to allow that some points are not covered by any sensor. This might be the case where we use grid approach to approximate area coverage. Accordingly, some other variants of sensor placement problems include (a) maximizing the number of covered targets subject to a given number of sensors and (b) minimizing the network cost subject to a minimum target coverage ratio. For directional coverage model, another objective is to optimize sensor orientational angles to maximize target coverage [7]. All these types of sensor placement problems are basically optimization problems and can be formulated as various mathematical programming problems.

4.2.2 Approximation Algorithms

For small problem instances (for example, the available sites for placing sensors are less than 20), exhaustive search can be used to find the global optimum by trying every possible placement. If we have I available sites, we can choose i sites each time for placing sensors and examine the cost of the placement. The number of all possible placements that need to be examined is $\sum_{i=1}^{I}\binom{I}{i} = 2^I - 1$. That is, the computation complexity for exhaustive search increases exponentially with the number of available sites. For the mathematical programming problems discussed in the previous subsection, some standard optimizer such as CPLEX [8] can be used to solve these problems with faster computation speed than the exhaustive search. However, it is worth noting that the computation is still a very time-consuming process, especially when the problem instance is large. When the field has a symmetry (e.g., sensors are to be placed at vertices of a 100×100 grid), a simple approximation method is to divide the field into several congruent subfields, solve the optimization problem for one subfield, and extend the result to the rest subfields. However, such a division may not be optimal. Therefore, the solution to the whole field may not be globally optimal, even if the solution to a subfield is optimal. Some well-known

approximation algorithms, such as greedy algorithm, simulated annealing, and genetic algorithms, etc., can be applied to find approximate solutions to the sensor placement problems.

Greedy Algorithm The GREEDY-SET-COVER algorithm introduced in the previous section is computation-efficient for large problem instances. Its computation runs in time $O(IJ \cdot \min\{I, J\})$, compared with 2^I of the exhaustive search. In general, an algorithm is called *greedy* if it divides the problem into subproblems to be solved in consecutive stages and always takes the best local solution at each stage. The *greedy* decision made in each stage may depend on the decisions made so far but not on future decisions. With such greedy decisions, the original problem are iteratively reduced to many smaller subproblems. A greedy algorithm terminates if a predefined optimal threshold has been achieved or the maximal allowable stages have been performed. Many variants of the simple GREEDY-SET-COVER algorithm have been proposed to solve various node placement problems [4–6, 19, 20, 22, 26]. For example, we can normalize the cost of placing a type a sensor at the site i by c_a/n_j. n_j is the number of uncovered targets that can be covered if a type a sensor is placed at site i. In the decision stage, the least normalized cost is selected, and then the site is allocated to the sensor type with the least normalized cost. The newly covered targets are then removed from the set of uncovered targets, and the process is terminated when all targets are covered.

Simulated Annealing Simulated annealing is a generic probabilistic heuristic for locating a good approximation to the global extremum of a given function. Simulated annealing has been applied to solve various node placement problems [11–13]. The term simulated annealing derives from a roughly analogous physical process of heating and then slowly cooling a substance to obtain a strong crystalline structure [10]. The basic idea of the simulated annealing method is as follows. At first, the system has an initial state with a corresponding system value, and we set an initial temperature. The process consists of two loops. In the outer loop, the temperature is reduced in each step until it reaches a predefined threshold. In the inner loop (at each given temperature), we slightly change the system state (e.g., by randomly moving a sensor to a new site) and then compare the new system value with the old one. If the difference is less than another threshold, we then accept this state (maybe, with some probability) and then enter the next temperature. If the difference is still large, we change the system state again. For the node placement problem, the initial state is to place nodes to all available sites. In each temperature cooling step, we remove one sensor node, and in each temperature cooling step, the best site to remove a sensor node is found by running the inner loop and using the cost function to determine the difference of the system value.

Genetic Algorithm Genetic algorithms are a class of probabilistic optimization algorithms to search or to approximate solutions for optimization problems. Inspired by biological evolutionary processes, genetic algorithms model and apply biological inheritance, mutation, selection, and crossover (recombination) in the search of

global optimal solution. Genetic algorithm has been widely used to solve the combinatorial and nonlinear optimization problems with complex constraints and has also been applied to find solutions to various sensor placement problems [16, 21, 23, 25]. The computation of genetic algorithm is an iterative process towards achieving the global optimality. During the iterations, candidate solutions are retained and ranked according to their quality. A fitness value is used to screen out unqualified solutions. Genetic operations of crossover, mutation, translocation, inversion, addition, and deletion are then performed on those qualified solutions to create new candidate solutions of the next generation. The above process is carried out repeatedly until certain stopping or convergence condition is met For simplicity, a maximum number of iterations can be chosen to be the stopping condition. The variation difference of the fitness values between two adjacent generations may also serve as a good indication for convergence.

4.2.3 Other Placement Problems

Discriminative Coverage In the sensing disk coverage model, a target j is said to be covered by a sensor i if its Euclidean distance to the sensor is not larger than the radius of the sensor disk, i.e., $d(i, j) \leq R_s$. Also the sensing disk is often modeled as a binary coverage model (or a Boolean coverage model): A target j is either covered by a sensor i ($\delta_{ij} = 1$ when $d(i, j) \leq R_s$) or not ($\delta_{ij} = 0$ when $d(i, j) > R_s$). A single target can be covered by more than one sensor, and a single sensor can also cover more than one target. For a single target j, we can construct a *coverage vector* to describe its coverage by a set of sensors. For example, as shown in Fig. 4.3, there are six sensor nodes located at the grid points 4, 6, 7, 9, 10, and 12, and the grid point 8 is covered by sensors located at grid points 7 and 9. The corresponding coverage vector for the grid point 8 is $V_8 = (0, 0, 1, 1, 0, 0)$. In a sensor network, if all targets are covered by at least one sensor, then this sensor network is said to provide complete coverage for all targets. Furthermore, if each target is covered by at least one sensor and can be identified by a unique coverage vector, then this network is said to provide complete discriminative coverage for these targets. In Fig. 4.3, each grid point is covered by one sensor and has a unique coverage vector. This complete discriminative coverage actually indicates that the minimum Hamming distance of the coverage vectors associated with any pair of target points is at least one. The Hamming distance between two coverage vectors is the number of positions for which the corresponding values are different. For example, in Fig. 4.3, the coverage vector for the grid point 7 is $V_7 = (0, 1, 1, 0, 0, 1)$, and the Hamming distance between V_7 and V_8 is 3.

Lin and Chiu [12] argue that the requirement of providing complete discriminative coverage might be too strict. Instead, they formulate the sensor placement problem as a combinatorial optimization problem where the objective is to achieve high discrimination among targets other than complete discrimination. The objec-

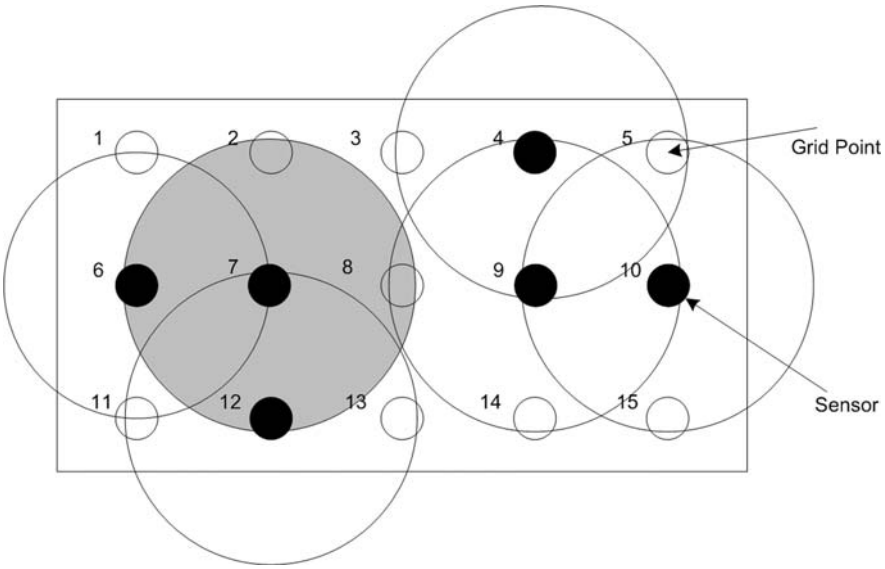

Fig. 4.3 An illustration of a sensor network providing complete and discriminative coverage for all grid vertices

tive is expressed as

$$\text{Minimize} \quad \max_{(i_1,i_2)} \frac{d(i_1,i_2)}{1 + C\sum_{j=1}^{J}(V_{i_1 j} - V_{i_2 j})^2},$$

where $d(i_1, i_2)$ is the Euclidean distance between two targets i_1 and i_2, $V_{i_1 j}$ and $V_{i_2 j}$ are the coverage vectors, and C is a large constant. Obviously, the larger the Hamming distance between two coverage vectors, the larger the value of $\sum(V_{i_1 j} - V_{i_2 j})^2$, and the smaller value of the cost function. Lin and Chiu [12] propose to apply the *simulated annealing* technique to find a near-optimal node placement minimizing the cost function.

Detection Coverage Some node placement problems are based on probabilistic sensing and detection coverage models [4, 5, 17, 18, 24, 26]. In the detection coverage model, a target j is assigned a detection probability ($0 \le p_{ij} \le 1$) which measures its detectability by a sensor i. The detection probability dependent on the distance between the target and the sensor. For example, a simple exponential detection model [5] is as follows:

$$p_{ij} = e^{-\alpha d(i,j)},$$

where α is the exponent coefficient, and $d(i, j)$ is the Euclidean distance between target j and sensor i. Another detection model considers two cut-off detection

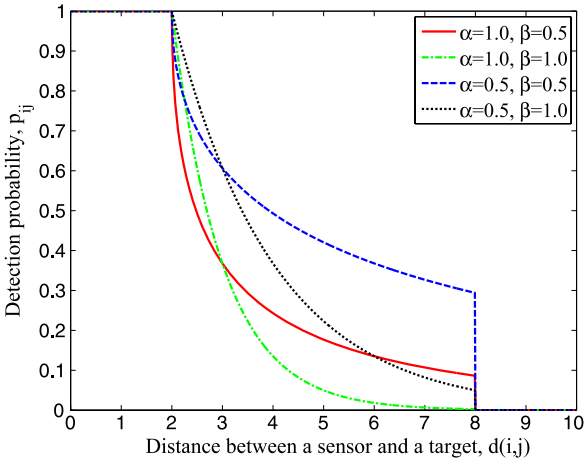

Fig. 4.4 Illustration of the detection model with two cut-off thresholds (for the color version, see Color Plates on p. 209)

thresholds [26]:

$$p_{ij} = \begin{cases} 1 & \text{if } d(i,j) \le r - r_e, \\ e^{-\alpha a^{\beta}} & \text{if } r - r_e < d(i,j) < r + r_e, \\ 0 & \text{if } d(i,j) > r + r_e, \end{cases}$$

where r_e ($r_e < r$) is a measure of the uncertainty in sensor detection, $a = d(i,j) - (r - r_e)$, and α and β are detection parameters. Figure 4.4 illustrates such a detection model with different choices of the parameters.

Assuming that the detection of each sensor is independent of the others, the overall detection probability for a single target j by I sensors can be computed by

$$p_j = 1 - \prod_{i=1}^{I} (1 - p_{ij}).$$

Different coverage requirements can be applied when placing sensors. For example, the overall detection probability should be larger than a predefined threshold for all targets, i.e., $p_j \ge p^*$, $j = 1, \ldots, J$. The objective is to place the minimum number of sensors to achieve the detection coverage requirements. Similar to the set-covering problem, this sensor placing problem can also be approximated by the greedy algorithm but with some modifications. In the decision step, a sensor is placed at the site such that the network-wide detection probability due to this placement, that is, $\sum_{j=1}^{J} p_{ij}$, is maximized. Since the overall detection probability depends on all the deployed sensors, it needs to be updated after the decision for a newly deployed sensor. The selection loop terminates if all targets achieve the coverage requirement or the maximum allowable number of nodes have been placed.

References

1. Altinel, K., Aras, N., Güney, E., Ersoy, C.: Binary integer programming formulation and heuristics for differentiated coverage in heterogeneous sensor networks. Computer Networks (Elsevier) **52**(12), 2419–2431 (2008)
2. Chakrabarty, K., Iyengar, S.S., Qi, H., Cho, E.: Grid coverage for surveillance and target location in distributed sensor networks. IEEE Transactions on Computers **51**(12), 1448–1453 (2002)
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd Edition. MIT Press, Cambridge (2001)
4. Dhillon, S.S., Chakrabarty, K.: Sensor placement for effective coverage and surveillance in distributed sensor networks. In: IEEE Wireless Communications and Networking Conference (WCNC), pp. 1609–1614 (2003)
5. Dhillon, S.S., Chakrabarty, K., Iyengar, S.: Sensor placement for grid coverage under imprecise detections. In: IEEE International Conference on Information Fusion, pp. 1581–1587 (2002)
6. Fang, Z., Wang, J.: Convex combination approximation for the min-cost WSN point coverage problem. In: The 3rd International Conference on Wireless Algorithms, Systems, and Applications (WASA), also in LNCS, vol. 5258, pp. 188–199 (2008)
7. Hörster, E., Lienhart, R.: On the optimal placement of multiple visual sensors. In: ACM the 4th International Workshop on Video Surveillance and Sensor Networks (VSSN), pp. 111–120 (2006)
8. ILOG Inc., I.: Using the CPLEX callable library. http://cplex.ilog.com (2009)
9. Ke, W.C., Liu, B.H., Tsai, M.J.: Constructing a wireless sensor network to fully cover critical grids by deploying minimum sensors on grid points is NP-complete. IEEE Transactions on Computers **56**(5), 710–715 (2007)
10. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
11. Lin, F.Y.S., Chiu, P.L.: Energy-efficient sensor network design subject to complete coverage and discrimination constraints. In: IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON) (2005)
12. Lin, F.Y.S., Chiu, P.L.: A near-optimal sensor placement algorithm to achieve complete coverage-discrimination in sensor networks. IEEE Communications Letters **9**(1), 43–45 (2005)
13. Lin, F.Y.S., Chiu, P.L.: A simulated annealing algorithm for energy-efficient sensor network design. In: International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), pp. 183–189 (2005)
14. Patel, M., Chandrasekaran, R., Venkatesan, S.: Energy efficient sensor, relay and base station placements for coverage, connectivity and routing. In: IEEE International Performance, Computing and Communications Conference (IPCCC), pp. 581–586 (2005)
15. Sen, A., Das, N., Zhou, L., Shen, B.H., Murthy, S., Bhattacharya, P.: Coverage problem for sensors embedded in temperature sensitive environments. In: IEEE the 4th Annual Communications Society on Sensor and Ad Hoc Communications and Networks (SECON), pp. 1–10 (2007)
16. Seo, J.H., Kim, Y.H., Ryou, H.B., Cha, S.H., Jo, M.: Optimal sensor deployment for wireless surveillance sensor networks by a hybrid steady-state genetic algorithm. IEICE Transactions on Communications **E91-B**, 3534–3543 (2008)
17. Stolkin, R., Florescu, I.: Probability of detection and optimal sensor placement for threshold based detection systems. IEEE Sensor Journal **9**(1), 57–60 (2009)
18. Stolkin, R., Vickers, L., Nickerson, J.V.: Using environment models to optimize sensor placement. IEEE Sensor Journal **7**(3), 319–320 (2007)
19. Wang, B.: Sensor placement for complete information coverage in distributed sensor networks. Journal of Circuits, Systems, and Computers (World Scientific) **17**(4), 627–636 (2008)

20. Wang, J., Zhong, N.: Efficient point coverage in wireless sensor networks. Journal of Combinatorial Optimization **11**(3), 291–304 (2006)
21. Wu, Q., Rao, N.S., Du, X., Iyengar, S.S., Vaishnavi, V.K.: On efficient deployment of sensor on planar grid. Computer Communications (Elsevier) **30**(14–15), 2721–2734 (2007)
22. Xu, X., Sahni, S.: Approximation algorithms for sensor deployment. IEEE Transactions on Computers **56**(12), 1681–1695 (2007)
23. Xu, Y., Yao, X.: A GA approach to the optimal placement of sensors in wireless sensor networks with obstacles and preferences. In: IEEE 3rd Consumer Communications and Networking Conference (CCNC), pp. 127–131 (2006)
24. Zhang, J., Yan, T., Son, S.H.: Deployment strategies for differentiated detection in wireless sensor networks. In: IEEE the 3rd Communications Society on Sensor and Ad Hoc Communications and Networks (SECON), pp. 316–325 (2006)
25. Zhao, C., Yu, Z., Chen, P.: Optimal deployment of nodes based on genetic algorithm in heterogeneous sensor networks. In: IEEE International Conference on Wireless Communications, Networking and Mobile Computing (WiCom), pp. 2743–2746 (2007)
26. Zou, Y., Chakrabarty, K.: Uncertainty-aware and coverage-oriented deployment for sensor networks. Journal of Parallel and Distributed Computing **64**(7), 788–798 (2004)

Chapter 5
Coverage Lifetime Maximization

Abstract In remote or hostile sensor fields, randomly scattering sensor nodes (e.g., dropping from an aircraft) might be the only way to deploy sensor networks. If only one sensor is air-dropped at the proximity of a target, it is possible that this target is not within the sensing range of this sensor due to the randomness. In order to increase the likelihood of target coverage, it is often to disperse many sensor nodes around each target. In such a randomly deployed sensor network, a target may be covered by more than one sensor, and a sensor may also cover more than one target. All sensor nodes are assumed to be equipped with only limited energy supply (e.g., two AAA batteries), and hence the network operational time is not unlimited. It is not wise to switch on all sensors as this consumes the sensor energy fast and results in a short-lived network. We can partition sensors into different *covers* (each a subset of sensors that can satisfy the coverage requirement) and activate these covers in a round-robin fashion. The network operating time for target coverage is then the total time span of these sensor covers' runtime. The objective of the coverage lifetime maximization can be summarized as to answer the following question:

How to partition sensors into different sensor covers and schedule their operating intervals so that the coverage requirement can be satisfied by these covers and the target coverage lifetime can be maximized?

Depending on the energy consumption models and target coverage requirements, many variants of this coverage lifetime maximization problem have been proposed and studied in the literature. This chapter introduces the mathematical models and solution approaches for these coverage lifetime maximization problems.

5.1 Maximizing Target Coverage Lifetime

Let us first consider a simple example of target coverage. As illustrated in Fig. 5.1(a), there are six sensors and four targets in a randomly deployed network. If we consider a disk coverage model, then the targets z_2 and z_4 are each covered by two sensors; and the targets z_1 and z_3 are each covered by three sensors. We use *coverage mapping* to refer to the coverage relations among all sensors and all

B. Wang, *Coverage Control in Sensor Networks,*
Computer Communications and Networks,
DOI 10.1007/978-1-84996-059-5_5, © Springer-Verlag London Limited 2010

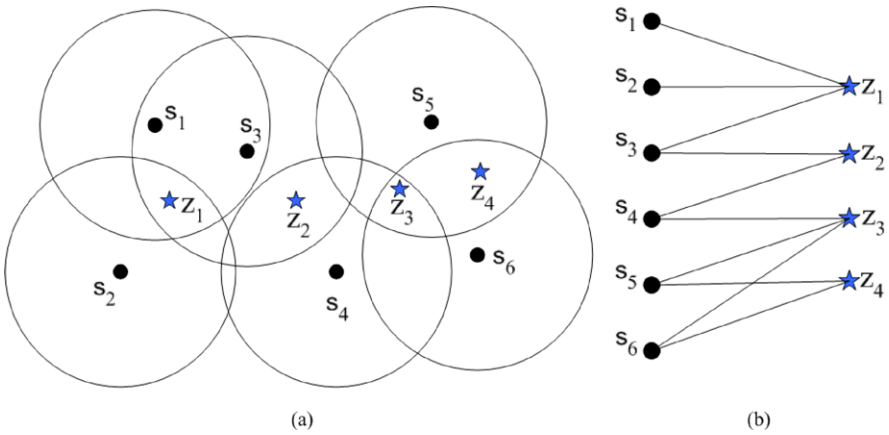

Fig. 5.1 Illustration of (**a**) a randomly deployed sensor network for covering targets and (**b**) the corresponding sensor-target bipartite graph

targets that can be represented by a *sensor-target bipartite graph*: The vertices are the sensors and targets, and an edge exists between a sensor and a target if the sensor covers the target. Figure 5.1(b) plots the sensor-target bipartite graph of the example sensor network.

In the context of target coverage, it is a common prerequisite that all targets can be covered if all sensors are activated for sensing. However, activating all the sensors at the same time is not energy efficient. If every sensor can only operate for one time unit in a continuously active state, then activating all sensors all the time results in a total network lifetime of also one time unit. Instead, we can alternatively activate sensors. For example, in the network shown in Fig. 5.1, we can activate $C_1 = \{s_1, s_3, s_6\}$ for one time unit and $C_2 = \{s_2, s_4, s_5\}$ for another time unit. Since all targets are still covered by either C_1 or C_2, the coverage requirements are not sacrificed. Furthermore, the target coverage lifetime can be extended to two time units. Obviously, we can have other choices of partitioning the sensors into different subsets, such as $C_3 = \{s_1, s_2, s_5\}$ and $C_4 = \{s_3, s_6\}$. The objective of the target coverage problem is to find the optimal subsets and their active intervals such that the coverage requirements can be satisfied and the total target coverage lifetime can be maximized.

We next provide a generalized yet simplified model for the target coverage lifetime maximization problem. The symbols used in this chapter are summarized in Table 5.1. We consider a sensor network of N sensors and M targets. In what follows, we use the subscript i and j to index sensors and targets, respectively. Let $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$ and $\mathcal{Z} = \{z_1, z_2, \ldots, z_M\}$ denote the sensor set and target set, respectively. For the sensing disk coverage model, it is commonly assumed that if a sensor is active, it can cover all targets within the sensing disk centered at itself with the radius of sensing range. A sensor may cover several targets, and a target might be covered by multiple sensors. We use $Z(s_i)$ to denote the set of targets covered by the sensor s_i, and its cardinality $|Z(s_i)|$ is the number of covered targets.

Table 5.1 The symbols and their notations

s_i	the ith sensor (indexed by subscript i, $i = 1, 2, \ldots, N$)
$\mathcal{S} = \{s_1, \ldots, s_N\}$	the set of all deployed sensors, $\lvert\mathcal{S}\rvert = N$
z_j	the jth target (indexed by subscript j, $j = 1, 2, \ldots, M$)
$\mathcal{Z} = \{z_1, \ldots, z_M\}$	the set of all targets, $\lvert\mathcal{Z}\rvert = M$
$S(z_j)$	the set of sensors that can cover target z_j
$Z(s_i)$	the set of targets covered by sensor s_i
$\mathcal{C} = \{Z(s_1), \ldots, Z(s_N)\}$	the collection of targets covered by each sensor
C_k	the kth set cover (indexed by subscript k, $k = 1, \ldots, K$)
$\lvert C_k\rvert$	the number of sensors in the set cover C_k
$\{C_k\}$	the set of targets covered by the set cover C_k
$\lvert\{C_k\}\rvert$	the number of targets covered by the set cover C_k
$\delta(s_i, C_k)$	the inclusion function whether s_i is in C_k: $\delta(s_i, C_k) = 1$ if $s_i \in C_k$; otherwise, $\delta(s_i, C_k) = 0$
$\psi(z_j, C_k)$	the coverage mapping between z_j and C_k: $\psi(z_j, C_k) > 0$ if z_j is covered by at least one sensor in C_k; otherwise, $\psi(z_j, C_k) = 0$
$\mathcal{R}$	the sink
$\mathcal{T}_k$	the kth cover tree (indexed by subscript k, $k = 1, \ldots, K$)
t_k	the active time interval for set cover C_k
T	the network lifetime for target coverage
E_i	the initial energy of sensor s_i
E_i^r	the residual energy of sensor s_i
$e_k(s_i, C_k)/e_k(s_i, \mathcal{T}_k)$	the energy consumption of sensor s_i when only $C_k/\mathcal{T}_k$ is working for t_k time units
$B(t)$	the sensing data produced by a sensor within t time units

We use $S(z_j)$ to denote the set of sensors covering the target z_j, and its cardinality $\lvert S(z_j)\rvert$ is the number of sensors covering the target z_j. For example, in Fig. 5.1, $Z(s_3) = \{z_1, z_2\}$, $\lvert Z(s_3)\rvert = 2$ and $S(z_3) = \{s_4, s_5, s_6\}$, $\lvert S(z_3)\rvert = 3$.

We use $\mathcal{C}$ to denote a collection of subsets of $\mathcal{Z}$. The ith element in $\mathcal{C}$ is the target covered by the sensor s_i, that is, $\mathcal{C} = \{Z(s_1), \ldots, Z(s_N)\}$. For example, in Fig. 5.1, $\mathcal{C} = \{\{z_1\}, \{z_1\}, \{z_1, z_2\}, \{z_2, z_3\}, \{z_4, z_5\}, \{z_4, z_5\}\}$. We call a subset of sensors a (sensor) *set cover*, if it can cover at least one target. We use the subscript k to denote the kth set cover C_k. The cardinality of the set cover C_k, $\lvert C_k\rvert$, is the number of sensors in C_k. We use an *inclusion indicator function* $\delta(s_i, C_k)$ to indicate whether the sensor s_i is included in the set cover C_k. That is, $\delta(s_i, C_k) = 1$ if $s_i \in C_k$ and $\delta(s_i, C_k) = 0$ otherwise. With a little abuse of symbols, we use $\{C_k\}$ to denote the set of the targets covered by a set cover C_k and use $\lvert\{C_k\}\rvert$ to denote the number of covered targets. For example, in Fig. 5.1, the set cover C_1 consists of sensors s_1, s_3, and s_6, and its cardinality $\lvert C_1\rvert = 3$. The set of targets covered by C_1 is $\{C_1\} = \{z_1, z_2, z_3, z_4\}$, and $\lvert\{C_1\}\rvert = 4$.

We assume that the coverage mapping between sensors and targets are known. We use $\psi(z_j, C_k) \geq 0$ to denote the coverage mapping function between a target z_j and a set cover C_k. C_k can be a set containing only a single sensor or containing several sensors. $\psi(z_j, C_k) = 0$ means that the target z_j is not covered by C_k, and $\psi(z_j, C_k) > 0$ means that is covered. A set cover is constructed to satisfy a predefined coverage requirement which is application-dependent and considered as an input to the lifetime maximization problem. For example, the complete target coverage requirement asks that a set cover C_k should cover all targets, that is, $\psi(z_j, C_k) > 0$ for all $z_j \in \mathcal{Z}$. A set cover is called *irreducible* if the coverage requirement cannot be satisfied any more by removing any sensor in the set cover. For example, in the above example, the set cover $C_1 = \{s_1, s_3, s_6\}$ is irreducible for the complete target coverage requirement.

We can extend network lifetime by alternatively activating sensors. This corresponds to construct a series of sensor covers, C_k, and allocate each operating time interval $t_k, t_k > 0$, $k = 1, 2, \ldots$. All the sensors in C_k are activated for covering targets in the allocated interval t_k, and the sensors not in C_k deactivate their sensing unit to save their energy. Let $e_k(s_i, C_k) \geq 0$ denote the energy consumption of a sensor s_i when only the set cover C_k is selected to be active for t_k time units. $e_k(s_i, C_k) = 0$ indicates that s_i consumes no energy, which implies that $s_i \notin C_k$. $e_k(s_i, C_k)$ does depend on the role of the sensor in the interval t_k and the detailed energy consumption model in different application scenarios. Furthermore, every sensor in C_k cannot consume the energy more than its residual energy in a time interval t_k, which implicitly restricts the length of an active interval. Each sensor is assumed to have limited initial energy supply E_i, $i = 1, 2, \ldots, N$. A sensor is dead if its residual energy becomes zero. An irreducible set cover containing a dead sensor cannot satisfy the coverage requirement and is no longer used for target coverage.

The *target coverage lifetime* is defined as the duration from the time that the network starts operation till the time that the coverage requirement cannot be satisfied even if all sensors are activated. Since all sensor nodes have only limited initial energy, the network lifetime for target coverage hence is also limited. The objective of the coverage lifetime maximization problem is to find an optimal schedule (C_k, t_k), $k = 1, 2, \ldots$, consisting of the set covers C_k and their corresponding active intervals t_k, such that the coverage requirement can be satisfied by each set cover during its operating interval, and the coverage lifetime can be maximized. We use $\mathbf{\Psi}(\mathcal{Z}, C_k)$ as an indicator function to denote whether the network-wide target coverage requirement can be satisfied by a set cover C_k. That is, if coverage requirement is satisfied, then $\mathbf{\Psi}(\mathcal{Z}, C_k) = 1$, and otherwise, $\mathbf{\Psi}(\mathcal{Z}, C_k) = 0$. The coverage lifetime maximization problem can be formulated as the following optimization problem. The lifetime T is defined as the sum of all the active intervals, (5.1). The number of set covers is denoted by K. Since all nodes have limited energy, both t_k and K are finite but unknown. Two basic constraints are the energy constraint, (5.2), and the coverage requirement, (5.3).

Coverage Lifetime Maximization Problem

$$\text{Maximize:} \quad T \equiv \sum_{k=1}^{K} t_k \tag{5.1}$$

$$\text{Subject to:} \quad \sum_{k=1}^{K} e_k(s_i, C_k) \leq E_i \quad \text{for all } s_i \quad \text{(energy constraint)} \tag{5.2}$$

$$\Psi(\mathcal{Z}, C_k) = 1 \quad \text{for all } C_k \quad \text{(coverage constraint)} \tag{5.3}$$

$$\ldots \quad \text{(other constraints)}$$

The coverage requirements are diverse for different applications. There may have some other constraints depending on the assumptions and objectives of the problem. For example, it might be required that the sensor nodes in each sensor cover can also form a connected network to the sink. The lifetime maximization problem with connectivity constraint will be discussed in the next section. In the rest of this section, we introduce several examples of target coverage lifetime maximization without considering the constraint of connectivity among active nodes.

The lifetime maximization problem is much simplified if connectivity is not considered as a constraint. One scenario for not considering connectivity among nodes might be like this. The sensor field is farther away, and an aircraft as the sink flies over the sensor field to collect sensors' data. All active nodes send their sensed data directly to the sink, and they consume almost the same amount of energy for sending the data. In such an example, the energy consumption model is simple. All active sensor nodes consume the same amount energy per time unit; and all sleep sensor nodes do not consume any energy. However, depending on the coverage requirements, this lifetime maximization problem can be as hard as an NP-complete problem.

5.1.1 Disjoint Set Cover

We first introduce a kind of *disjoint set cover* (DSC) problem where the available sensors are partitioned into *disjoint* subsets to be activated consecutively. Two sets are called disjoint if their intersection is an empty set, that is, $C_k \cap C_{k'} = \emptyset$ for $k \neq k'$. All sensors are assumed to have the same amount of initial energy and have the same energy consumption rate in the active state. If a set cover is scheduled to be continuously active, then all sensors in the set cover will die at the same time. Each disjoint set cover is activated till its death, and all the disjoint set covers are activated one by one. With such an arrangement, the target coverage lifetime equals to the number of these set covers times the runtime of a single set cover. The objective of the lifetime maximization problem then can be converted to the problem of finding the maximal number of disjoint set covers that satisfy the coverage requirements.

Maximal Disjoint Set Cover for Complete Target Coverage A straightforward coverage requirement is the complete target coverage. All targets should be covered all the time. The complete coverage requirement indicates that $\Psi(\mathcal{Z}, C_k) = 1$ if $\{C_k\} = \mathcal{Z}$ and $\Psi(\mathcal{Z}, C_k) = 0$ otherwise. The objective is to find the maximal number of disjoint set covers, each covering all targets. The maximal disjoint set cover problem is defined as follows.

Definition 5.1 (Maximal Disjoint Set Cover (MDSC) Problem) Given a collection $\mathcal{C}$ of subsets of a finite set $\mathcal{Z}$, find the maximum number of disjoint set covers, C_k, $k = 1, 2, \ldots, K$, such that each set cover can cover all targets, $\{C_k\} = \mathcal{Z}$ for all C_k, and for any two covers C_k and $C_{k'}$, $C_k \cap C_{k'} = \emptyset$.

The decision version of the MDSC problem is as follows: Given a collection $\mathcal{C}$ of subsets of a finite set $\mathcal{Z}$ and a positive integer K, can we find K disjoint set covers, each covering all targets? The MDSC problem is an NP-complete problem [9]. As a variant of the coverage lifetime maximization problem, the optimization version of the MDSC problem can also be formulated as the following maximization problem:

$$
\begin{array}{lll}
\text{Maximize:} & T \equiv K & \\
\text{Subject to:} & \sum_{k=1}^{K} \delta(s_i, C_k) \leq 1 \quad \text{for all } s_i & \text{(energy constraint)} \\
& \{C_k\} = \mathcal{Z} \quad \text{for all } C_k & \text{(coverage constraint)} \\
& C_k \cap C_{k'} = \emptyset \quad \text{for all } k \neq k' & \text{(disjoint constraint)} \\
& \delta(s_i, C_k) \in \{0, 1\} \quad \text{for all } s_i, C_k & \text{(inclusion constraint)}
\end{array}
$$

The network lifetime is defined by the number of disjoint set covers. This is because each set cover is active for one time unit. The energy constraint states that a sensor can be in at most one of the set covers. The coverage constraint requires complete target coverage. The inclusion indicator function $\delta(s_i, C_k)$ indicates whether the sensor s_i is included in the set cover C_k, that is, $\delta(s_i, C_k) = 1$ if $s_i \in C_k$ and $\delta(s_i, C_k) = 0$ otherwise.

An intuitive upper bound of K is determined by the minimal number of sensors covering a target, that is, $K \leq \min_{z_j \in \mathcal{Z}}(|S(z_j)|)$. This is because each sensor can only be in one of the set cover (disjoint constraint), and all targets should be covered by each set cover (coverage constraint). Recall that we use $S(z_j)$ to denote the set of the sensors covering the target z_j. Such a target is called a *critical target*, denoted by z^c, and $z^c = \arg\min_{z_j \in \mathcal{Z}}(|S(z_j)|)$ and hence $K \leq |S(z^c)|$. Cardei and Du [9] provide a tight bound for K. A flow network is first constructed, and the maximal flow which is obtained by solving a mixed integer program provides the upper bound of K.

The steps for constructing a flow network are as follows.

Step 1. Consider a bipartite directed graph $G = (\mathcal{S} \cup \mathcal{Z}, E)$. An edge $\overline{sz}$ exists if and only if the target z is covered by the sensor s, and each edge is assigned a

capacity $c_{sz} = 1$. Create a vertex X and connect every vertex z in $\mathcal{Z}$ to X with an edge of capacity 1.

Step 2. Find a critical target z^c, and let $L = |S(z^c)|$. Draw L copies of G, namely $G_1, G_2, \ldots, G_L$. In these L copies, let the first index in a vertex notation reflect the component it belongs to, e.g., a vertex s_i in G, is named $s_{1i}, s_{2i}, \ldots, s_{Li}$ in $G_1, G_2, \ldots, G_L$.

Step 3. Create a source node S, and for each s_i in $\mathcal{S}$, create a vertex s_{0i}. Then connect the source S with s_{0i} with an edge of capacity equal with the degree of s_i in G. Also, connect s_{0i} with s_{li} for any $1 \le l \le L$, and assign a capacity equal with the degree of s_i in G.

Step 4. Create two sinks Y_1 and Y_2. Connect each vertex X_l, $1 \le l \le L$, to Y_2 and assign a capacity M. Then connect every vertex z_{lj} with $1 \le l \le L$ and $1 \le j \le M$ to Y_1 and assign the capacity N.

Figure 5.2 provides an example of a sensor-target bipartite graph and the constructed flow network. The flow f is defined as an integer-valued function that satisfies the following properties:

P1. Flow constraint: for all $uv \in E$, $0 \le f_{uv} \le c_{uv}$. An additional condition to the classic flow network is that for any $v \ne Y_1$, $f_{uv} \in \{0, c_{uv}\}$.

P2. Flow conservation: for all $u \in V - \{S, Y_1, Y_2\}$, $\sum_{v \in V, uv \in E \text{ or } vu \in E} f_{uv} = 0$.

The objective of this maximum-flow problem is to maximize the flow received in Y_2. The following theorem provides a bound of L by relating the maximum-flow to the MDSC problem.

Theorem 5.1 (Cardei and Du [9], Theorem 3) *Given a collection $\mathcal{C}$ of subsets of a finite set $\mathcal{Z} = \{z_1, z_2, \ldots, z_M\}$, the MDSC problem returns c^* covers if and only if the maximum-flow problem obtains the flow c^*M in Y_2.*

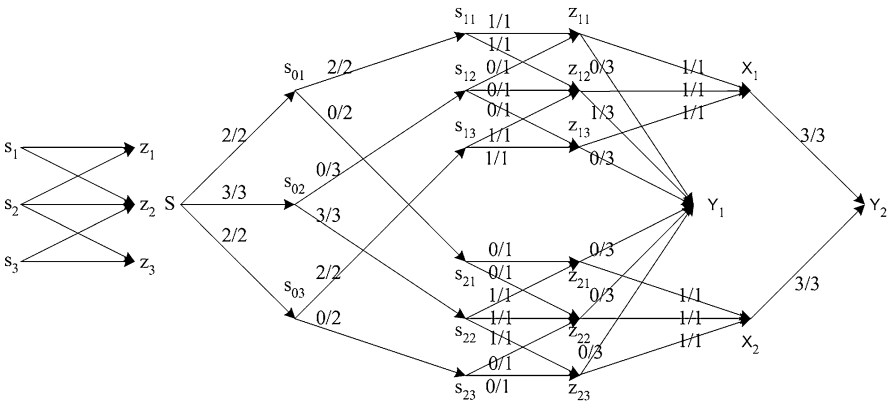

Fig. 5.2 Illustration of (**a**) a sensor-target bipartite graph, and (**b**) the constructed flow-network with flow/capacity assignment

Table 5.2 Pseudo-codes for the SP-heuristic [31]

```
01  Given S, Z, and set k = 1
02  while (all targets can be covered by S)
03    C_k = ∅; Z_uncvd = Z; S_left = S
04    while (Z_uncvd ≠ ∅)
05      find z^c = arg min_{z∈Z_uncvd} (|S(z)|);
06      max = −MAXINT
07      for each s_i ∈ S_left such that z^c ∈ Z(s_i)
08        f(s_i) = 0;
09        for each z ∈ Z(s_i)
10          if z ∈ Z_uncvd
11          then f(s_i) = f(s_i) + a_1 − a_2 × (|S(z) ∩ S_left|)
12          else f(s_i) = f(s_i) − a_3 + a_4 × (|S(z) ∩ (S − C_k)|) endif
13        endfor
14        if f(s_i) > max, then s* = s_i; max = f(s_i); endif
15        S_left = S_left − s*
16      endfor
17      C_i = C_i ∪ s*
18      Z_uncvd = Z_uncvd − Z(s*)
19    endwhile
20  S = S − C_i; i = i + 1
21  endwhile
```

Many algorithms can be used to solve the maximum-flow problem. The maximum-flow problem can also be formulated as a linear programming problem, and the optimal solution can be found by solving such a linear programming problem [9]. However, its computation is time consuming, especially for large problem instances. The transversion of the MDSC problem to the maximum-flow problem does not change its NP-completeness. On the other hand, approximation algorithms, such as greedy algorithm, simulated annealing, etc., can be applied to find disjoint set covers efficiently. For example, Slijepcevic and Potkonjak [31] propose a modified greedy algorithm (we call it SP-heuristic) for the MDSC problem, which greedily selects a sensor with a maximal profit to construct a set cover in each stage.

The pseudo-codes of the SP-heuristic is shown in Table 5.2, which consists of two main loops. In the outer loop (line 02–line 21), a set cover is constructed in each pass; The outer loop terminates if a target cannot be covered by the unselected sensors. In the inner loop (line 04–line 19), a sensor with a maximal profit is selected to construct a set cover. The inner loop terminates if all targets can be covered by the selected sensors. A critical target is defined as the one with the minimal number of sensors covering it. That is, $z^c = \arg\min_{z \in \mathcal{Z}}(|S(z)|)$. At the beginning of each pass of the inner loop (line 05), the critical target is first determined. Among the available sensors covering z^c, the one with the maximal profit is selected (line 06–line 15).

The profit function is defined as

$$f(s_i) = \sum_{z \in Z(s_i), z \in \mathcal{Z}_{\text{uncvd}}} \Big(\underbrace{a_1}_{(1)} - \underbrace{a_2 \times \big(\big|S(z) \cap \mathcal{C}_{\text{left}}\big|\big)}_{(2)} \Big) - \sum_{z \in Z(s_i), z \notin \mathcal{Z}_{\text{uncvd}}} \Big(\underbrace{a_3}_{(3)} - \underbrace{a_4 \times \big(\big|S(z) \cap (\mathcal{C} - C_i)\big)\big|}_{(4)} \Big),$$

where a_1, a_2, a_3, and a_4 are constants. This profit function is to favor the sensor that (1) covers a high number of uncovered targets; (2) covers more sparsely covered targets; (3) does not cover the targets redundantly, and (4) redundantly covers the targets which are not sparsely covered.

Disjoint Set K-Cover for Minimum Coverage Breach Covering all the targets all the time is a strict coverage requirement. Sometimes, we can relax this exacting requirement of complete coverage by allowing some *coverage breach* [12]. If a target is not covered by any active sensor, then it is said to be *breached*. In other words, the coverage requirement becomes a *partial target coverage* requirement, and a set cover is allowed to cover only a fraction of targets. Instead of running till its death, a set cover can only be activated for a short time duration, and all set covers are alternatively activated. With such rotative activations of set covers, a target that is not covered in this round may be covered in the next round.

On the other hand, some constraint may need to impose on the constructed set covers. For example, the cardinality of each set cover should not be larger than W. Recall the problem scenario introduced before this subsection: The sensor field is farther away. An aircraft as the sink flies over the sensor field to collect sensors' data, and all active nodes send their sensing data directly to the sink. When the aircraft flies over the sensor field, there are only W time slots available each for one sensor's data transmission. The W time slots can be considered as a "bandwidth" constraint of a single TDMA (time-division multiple access) channel. If the number of active sensors are larger than W, some of them cannot send their data due to such bandwidth constraint. To the observer, there is no difference between "not being covered" and "being covered but not being reported" [12].

Instead of finding the maximum number of disjoint set cover for complete target coverage, the disjoint set cover problem with allowable coverage breach is often called *Disjoint Set K-Cover* problem where the number of disjoint set covers to be constructed is predefined (i.e., K is predefined, and K disjoint set covers are to be constructed), and the objective is to minimize the coverage breach [1, 12–14]. Recall that we use $|C_k|$ to denote the cardinality of the set cover C_k, i.e., $|C_k|$ is the number of sensors in C_k. The bandwidth constraint indicates that $|C_k| \leq W$. The bandwidth constraint is not imposed when $W \geq |\mathcal{Z}|$. Recall that we use $\{C_k\}$ to denote the set of the targets covered by a set cover C_k and use $|\{C_k\}|$ to denote the number of covered targets. Note that $\{C_k\} \subseteq \mathcal{Z}$. In the Disjoint Set K-Cover problem, the partial target coverage indicates that $\{C_k\} \subset \mathcal{Z}$ is allowable. The Disjoint Set K-Cover problem is defined as follows.

Definition 5.2 (Disjoint Set K-Cover Problem) Given a collection $\mathcal{C}$ of subsets of a finite set $\mathcal{Z}$ and a positive integer K, find K disjoint set covers $(C_k, k = 1, 2, \ldots, K)$ with the bandwidth constraint ($|C_k| \leq W$ for all C_k) and the disjoint set constraint ($C_k \cap C_{k'} = \emptyset$ for any two covers C_k and $C_{k'}$), and achieve one of the following objectives:

(1) (*average coverage breach minimization*) Minimize the average number of breached targets over all set covers. That is, minimize $\frac{\sum_{k=1}^{K}(|\mathcal{Z}|-|\{C_k\}|)}{K}$.
(2) (*maximum coverage breach minimization*) Minimize the maximal number of breached targets over all set covers. That is, minimize $\max_k(|\mathcal{Z}| - |\{C_k\}|)$.
(3) (*maximum breach time minimization*) Minimize the maximal breach time of any particular target. That is, minimize $\max_j(\sum_{k=1}^{K}(1 - \psi(z_j, C_k)))$, where $\psi(z_j, C_k)$ is the coverage mapping function between a target z_j and a set cover C_k. For the boolean coverage model, $\psi(z_j, C_k) = 1$ if z_j is covered by C_k and $\psi(z_j, C_k) = 0$ otherwise.

The decision version of the Disjoint Set K-Cover problem for average coverage breach minimization is as follows: Given a collection $\mathcal{C}$ of subsets of a finite set $\mathcal{Z}$, a positive integer K, a positive integer W, and a real number B, can we divide $\mathcal{C}$ into K disjoint subsets such that the average number of breached targets is not larger than B, and each subset has at most W sensors in it? The decision versions for the other two objectives can also be formulated similarly. The Disjoint Set K-Cover problems are NP-complete problems [12]. The Disjoint Set K-Cover problem can be considered as another expression of the coverage lifetime maximization problem. In the Disjoint Set K-Cover problem, the coverage lifetime equals to the predefined number of set covers K, and the objective is to minimize the coverage breach. The optimization version of the Disjoint Set K-Cover problem can be formulated as the following minimization problem:

$$
\begin{array}{lll}
\text{Minimize:} & \dfrac{1}{K}\displaystyle\sum_{k=1}^{K}\left(|\mathcal{Z}| - \left|\{C_k\}\right|\right) & \text{(average breach minimization)} \\
\text{Subject to:} & \displaystyle\sum_{k=1}^{K}\delta(s_i, C_k) \leq 1 \quad \text{for all } s_i & \text{(energy constraint)} \\
& \displaystyle\sum_{i=1}^{N}\delta(s_i, C_k) \leq W \text{ for all } C_k & \text{(bandwidth constraint)} \\
& C_k \cap C_{k'} = \emptyset \quad \text{for all } k \neq k' & \text{(disjoint constraint)} \\
& \delta(s_i, C_k) \in \{0, 1\} \quad \text{for all } s_i, C_k & \text{(inclusion constraint)}
\end{array}
$$

The energy constraint states that a sensor can be in at most one of the set covers. The coverage constraint requires complete target coverage. The inclusion indicator function $\delta(s_i, C_k)$ indicates whether the sensor s_i is included in the set cover C_k.

For the other two minimization objectives, we just need to change the objective function.

The above optimization version of the Disjoint Set K-Cover problem is a typical integer programming problem, as all variables are integers. It can be first converted to a linear programming problem, and some commercial solvers can be used to find the optimal solution. The optimal solution to the linear programming problem may be fractional and need to be rounded to integer solution [12]. The computation for the optimal solution is time consuming, and some approximation algorithms have been proposed to solve the Disjoint Set K-Cover problem.

We first introduce a very simple randomized algorithm for the Disjoint Set K-Cover problem. The randomized algorithm assigns each sensor to a cover randomly from a uniform distribution. In particular, a sensor s_i randomly chooses an integer k uniformly distributed from 1 to K and assigns itself to the cover C_k. The randomized algorithm requires no processing and is very easy to be implemented. We use ζ to denote the partition of the set covers constructed by a particular algorithm and $M(\zeta) = \sum_{k=1}^{K} |\{C_k\}|$ to denote the total number of covered targets by the constructed set covers. Obviously, the average coverage breach $\frac{1}{K}(\sum_{k=1}^{K} |\mathcal{Z}| - \sum_{k=1}^{K} |\{C_k\}|)$ is minimized if $M(\zeta)$ is maximized. The following theorem states the relation between $M(\zeta_{\text{rand}})$ of the randomized algorithm and $M(\zeta^*)$ of the optimal solution.

Theorem 5.2 (Abrams et al. [1], Theorem 1) *The expected value of $M(\zeta_{\text{rand}})$ is a $1 - \frac{1}{e}$ approximation to $OPT = M(\zeta^*)$, where ζ_{rand} is the partition of the set covers constructed by the randomized algorithm, and ζ^* is the partition used by the optimum solution.*

Proof For a target z_j, we will calculate $\mathbb{E}[l_j]$, the expected number of covers that cover z_j in the randomized algorithm. We use N_j to denote the number of sensors covering z_j. A cover will not cover z_j with probability $(1 - \frac{1}{k})^{N_j}$ because there are N_j sensors to be assigned and each has probability of $\frac{1}{k}$ of being assigned to a particular cover. The expected number of covers containing z_j hence is $k - k(1 - \frac{1}{k})^{N_j}$. So the total expected number of times that targets are covered by the partition is $\sum_{z_j} \mathbb{E}[l_j] = \sum_{z_j} (k - k(1 - \frac{1}{k})^{N_j})$.

Let l_j^* be the number of times that z_j is covered in the optimum solution. Then, $l_j^* \leq \min(K, N_j)$ because a target cannot be covered by more than K covers or by more than the number of sensors covering it. The expected number of times that targets are covered by the OPT algorithm is at most $\sum_{z_j} \min(K, N_j)$. To show that the over fraction $\frac{\sum_{z_j} \mathbb{E}[l_j]}{\sum_{z_j} \min(K, N_j)} \geq (1 - \frac{1}{e})$, we will show that for all z_j, $\frac{\mathbb{E}[l_j]}{\min(K, N_j)} \geq (1 - \frac{1}{e})$. There are two cases.

I. $K \leq N_j$. Then, $\frac{\mathbb{E}[l_j]}{\min(K, N_j)} = 1 - (1 - \frac{1}{k})^{N_j} \geq 1 - (1 - \frac{1}{K})^{K} \geq 1 - \frac{1}{e}$. The last inequality is due to the power series expansion of e^x, which shows that $(1 - \frac{1}{K})^{K} \leq \frac{1}{e}$.

II. $K > N_j$. We will show that the derivation of the ratio $\frac{\mathbb{E}[l_j]}{\min(K,N_j)}$ with respect to N_j is negative, implying that the ratio is smallest when $K = N_j$. We have $\frac{\partial}{\partial N_j}(\frac{K-K(1-\frac{1}{K})^{N_j}}{N_j}) = \frac{K[(1-\frac{1}{K})^{N_j}(1-\ln(1-\frac{1}{K})^{N_j})-1]}{N_j^2}$. This is negative iff

$$1 + \ln\left(1 - \frac{1}{K}\right)^{-N_j} < \left(1 - \frac{1}{K}\right)^{-N_j},$$

which is true due to the power series expansion $(1 + t < e^t,\ t \neq 0$ [26]). □

We next introduce how the greedy algorithms can be applied to solve the Disjoint Set K-Cover problem [1]. The pseudo-codes of this centralized greedy algorithm is shown in Table 5.3. First, all the K set covers are initialized as empty sets. In each stage, the algorithm greedily assigns the sensor s_i to a cover C_k that maximizes the weighted sum of uncovered targets, $\sum_{z_j:(z_j \in Z(s_i)) \wedge (z_j \notin \{C_k\})}(1 - \frac{1}{k})^{N_j - 1}$. Each target which is covered by s_i ($z_j \in Z(s_i)$) and has not been covered by the sensors in the set cover C_k ($z_j \notin \{C_k\}$) is assigned a weight of $(1 - \frac{1}{K})^{N_j - 1}$, where N_j is the number of sensors that cover the target z_j. In this algorithm, each sensor is assigned to the cover where it covers the largest possible number of uncovered targets, weighted according to how likely it is that the target will be covered in future iterations. Abrams et al. [1] has also proven that this algorithm gives a $1 - \frac{1}{e}$ approximation for the Disjoint Set K-Cover problem.

Some other modified greedy algorithms have also been proposed for the Disjoint Set K-Cover problem [1, 12]. Abrams et al. [1] have also proposed a distributed greedy algorithm and have proven that it is a $\frac{1}{2}$ approximation for the Set K-Cover problem. Cheng et al. [12] propose a greedy algorithm with the bandwidth constraint. At each step, the effective coverage of a sensor s_i in a cover C_k is first computed, which is the number of targets covered by s_i and not yet covered by other sensors in the cover C_k. That is, $EC(s_i, C_k) = \sum_{z_j:(z_j \in Z(s_i)) \wedge (z_j \notin \{C_k\})} 1$. At each stage, the sensor s_{i*} and the cover C_{k^*} with the largest effective coverage are selected. Deshpande et al. [13, 14] convert the Set K-Cover problem into a Max K-Cut problem in a hyper-graph. In the hyper-graph, the vertices are the sensors, and for each target, a hyper-edge is created to contain all the sensors covering the target. The Max K-Cut problem can be approached by a kind of semidefinite programming based algorithms [13, 14].

Table 5.3 Pseudo-codes of the centralized greedy algorithm [1]

01	Given $\mathcal{C}$, $\mathcal{Z}$, and K
02	Initialize $C_1 = \emptyset, C_2 = \emptyset, \ldots, C_K = \emptyset$
03	**for** $i = 1$ to N
04	find $k = \arg\max_k \sum_{z_j:(z_j \in Z(s_i)) \wedge (z_j \notin \{C_k\})}(1 - \frac{1}{k})^{N_j - 1}$
05	$C_k = C_k \cup s_i$ (assign s_i to the cover C_K)
16	**endfor**

5.1.2 Nondisjoint Set Cover

In this subsection, we consider a kind of *nondisjoint set cover* problem where the constructed set covers need not be disjoint. Again, we assume that all sensor nodes have the same amount of initial energy and have the same energy consumption rate in the active state. The lifetime of a single sensor is assumed as one time unit if it is activated all the time. In the context of disjoint set cover problem, each sensor can only be included into one set cover, and all sensors have the same length of the active interval. Therefore, the network lifetime depends on the number of constructed disjoint sensor set covers. However, if we relax the disjoint constraint such that a sensor can be included in more than one set cover and each set cover can be activated for less than one time unit, then the network lifetime may be extended.

The following example introduced by Cardei et al. [10] provides a good illustration of the network lifetime extension by nondisjoint set covers. As shown in Fig. 5.3, if we partition the sensors into two disjoint set covers, namely, $\{s_1, s_2\}$ and $\{s_3, s_4\}$, then the network lifetime is 2. On the other hand, if we relax the disjoint constraint, then we can partition the sensors into four nondisjoint set covers and activate them for different time intervals, namely, $\{s_1, s_2\}$ for 0.5 time unit, $\{s_2, s_3\}$ for 0.5 time unit, $\{s_1, s_3\}$ for 0.5 time unit, and $\{s_4\}$ for 1 time unit. This partition results in a network lifetime of 2.5, which is a 25% increase in network lifetime compared with the disjoint set cover solution.

Maximum Set Cover (MSC) for Complete Target Coverage The complete target coverage requires that all targets should be covered all the time. This indicates that $\{C_k\} = \mathcal{Z}$ for all set cover C_k. We use $e_k(s_i, C_k)$, $e_k(s_i, C_k) \geq 0$, to denote the energy consumption of sensor s_i when the set cover C_k operates for t_k time units. Note that $e_k(s_i, C_k) = 0$ implies $s_i \notin C_k$. The energy constraint indicates that $\sum_{k=1}^{K} e_k(s_i, C_k) \leq E_i$, where E_i is the initial energy of sensor s_i. The objective is to find a *schedule* (C_k, t_k), $k = 1, 2, \ldots$, of a series of set covers C_k and their active intervals t_k such that the sum of all active intervals $\sum_k t_k$ is maximized. The maximal set cover problem is defined as follows.

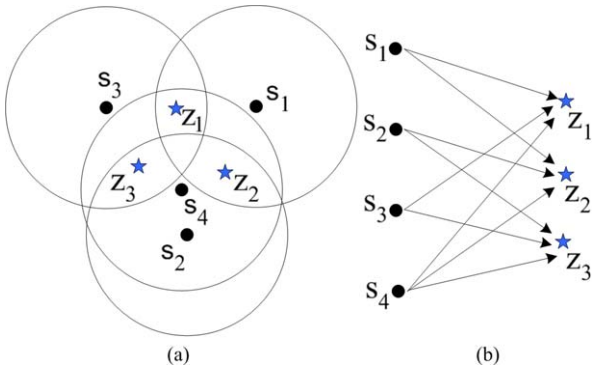

Fig. 5.3 Illustration of (**a**) a sensor network consisting of three sensors $\{s_1, s_2, s_3, s_4\}$ and four targets $\{z_1, z_2, z_3\}$ and (**b**) the corresponding sensor-target bipartite graph

Definition 5.3 (Maximum Set Cover (MSC) Problem) Given a collection $\mathcal{C}$ of subsets of a finite set $\mathcal{Z}$, find a schedule (C_k, t_k) of the set covers $C_1, C_2, \ldots, C_K$ with time intervals $t_1, t_2, \ldots, t_K$ to maximize $\sum_{k=1}^{K} t_k$ such that each set cover can cover all targets ($\{C_k\} = \mathcal{Z}$) for all C_k) and each sensor does not consume more than its initial energy ($\sum_{k=1}^{K} e_k(s_i, C_k) \leq E_i$ for all s_i).

The decision version of the MSC problem is as follows: Given a collection $\mathcal{C}$ of subsets of a finite set $\mathcal{Z}$ and a positive number T, can we find a family of set covers $C_1, \ldots, C_K$ with time intervals $t_1, t_2, \ldots, t_K$ such that $\sum_{k=1}^{K} t_k \geq T$, $\{C_k\} = \mathcal{Z}$ for all C_k, and $\sum_{k=1}^{K} e_k(s_i, C_k) \leq E_i$ for all s_i. The MSC problem is an NP-complete problem [10]. Let us first consider the simplest MSC problem where all sensors have the same lifetime of one time unit, and all targets need to be covered by at least one target at all the time. The optimization version of this MSC problem can be formulated as follows:

$$\begin{aligned}
\text{Maximize:} \quad & T \equiv \sum_{k=1}^{K} t_k \\
\text{Subject to:} \quad & \sum_{k=1}^{K} \delta(s_i, C_k) t_k \leq 1 \quad \text{for all } s_i \qquad \text{(energy constraint)} \\
& \{C_k\} = \mathcal{Z} \quad \text{for all } C_k \qquad \text{(coverage constraint)} \\
& \delta(s_i, C_k) \in \{0, 1\} \quad \text{for all } s_i, C_k \qquad \text{(inclusion constraint)}
\end{aligned}$$

The energy constraint states that each sensor cannot be activated for more than one time unit. The coverage constraint requires that each target should be covered by at least one sensor in a set cover. The inclusion indicator function $\delta(s_i, C_k)$ indicates whether the sensor s_i is included in the set cover C_k. That is, $\delta(s_i, C_k) = 1$ if $s_i \in C_k$ and $\delta(s_i, C_k) = 0$ otherwise.

In the above optimization problem, the term $\delta(s_i, C_k)t_k$ is not linear. We can converted it to a linear programming problem by setting $x_{ik} = \delta(s_i, C_k)t_k$, where $x_{ik} = 0$ or $x_{ik} = t_k \leq 1$. Furthermore, the complete coverage constraint is rephrased as $\sum_{i \in S(z_j)} x_{ik} \geq t_k$ for all $z_j \in \mathcal{Z}$. The linear programming (LP) formulation of the MSC problem is as follows:

$$\text{Maximize:} \quad T \equiv \sum_{k=1}^{K} t_k \tag{5.4}$$

$$\begin{aligned}
\text{Subject to:} \quad & \sum_{k=1}^{K} x_{ik} \leq 1 \quad \text{for all } s_i \qquad (5.5) \\
& \sum_{i \in S(z_j)} x_{ik} \geq t_k \quad \text{for all } z_j \in \mathcal{Z} \text{ and } k = 1, 2, \ldots, K \\
& 0 \leq x_{ik} \leq t_k \leq 1
\end{aligned}$$

The number of the set covers K, however, is unknown and needs to be preset in order to apply some LP solvers.

One approach to decide K is to apply an exhaustively search for all *irreducible* set covers [5, 6]. Recall that a set cover is called irreducible if the coverage requirement cannot be satisfied by removing any sensor from the set cover. The worst case of the search space can be as large as 2^N, where $N = |\mathcal{S}|$ is the total number of sensor nodes. For the complete coverage requirement, the worst case of the search space is upper bounded by $2^{N_z^{\max}}$, where $N_z^{\max} = \max_{z_j \in \mathcal{Z}} |S(z_j)|$ is the maximum number of sensors covering a target. We use $\mathcal{C}_{\text{all}}$ to denote the set of irreducible set covers in which each set cover can satisfy the coverage requirement and set $K_{\text{all}} = |\mathcal{C}_{\text{all}}|$. For the above LP problem, we can cancel the coverage constraint and rewrite it as

$$\begin{aligned} \text{Maximize:} \quad & T \equiv \sum_{k=1}^{K_{\text{all}}} t_k \\ \text{Subject to:} \quad & \sum_{k=1}^{K_{\text{all}}} x_{ik} \leq 1 \quad \text{for all } s_i \\ & x_{ik} \geq 0. \end{aligned}$$

This is a standard form LP problem, and the solution can be obtained via some commercial LP solver. We provide an example of using the simplex method to solve the MSC instance shown in Fig. 5.3.

Example 1 For the sensor network shown Fig. 5.3, the set of irreducible set covers for complete coverage is $\mathcal{C}_{\text{all}} = \{\{s_1, s_2\}, \{s_1, s_3\}, \{s_2, s_3\}, \{s_4\}\}$. The objective is to maximize the time intervals t_1, t_2, t_3, t_4 corresponding to the four set covers. The LP formulation is given by

$$\begin{aligned} \text{Maximize:} \quad & T \equiv t_1 + t_2 + t_3 + t_4 \\ \text{Subject to:} \quad & t_1 + t_2 \leq 1 \quad (s_1 \text{ energy constraint}) \\ & t_1 + t_3 \leq 1 \quad (s_2 \text{ energy constraint}) \\ & t_2 + t_3 \leq 1 \quad (s_3 \text{ energy constraint}) \\ & t_4 \leq 1 \quad (s_4 \text{ energy constraint}) \\ & t_1, t_2, t_3, t_4 \geq 0. \end{aligned}$$

t_1	t_2	t_3	t_4	τ_1	τ_2	τ_3	τ_4	T	
1	1	0	0	1	0	0	0	0	1
1	0	1	0	0	1	0	0	0	1
0	1	1	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1	0	1
−1	−1	−1	−1	0	0	0	0	1	0

$\Rightarrow$

t_1	t_2	t_3	t_4	τ_1	τ_2	τ_3	τ_4	T	
1	1	0	0	1	0	0	0	0	1
0	−1	1	0	−1	1	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1	0	1
0	0	−1	−1	1	0	0	0	1	1

$\Rightarrow$

t_1	t_2	t_3	t_4	τ_1	τ_2	τ_3	τ_4	T	
1	1	0	0	1	0	0	0	0	1
0	−1	1	0	−1	1	0	0	0	0
0	2	0	0	1	−1	1	0	0	1
0	0	0	1	0	0	0	1	0	1
0	−1	0	−1	0	1	0	0	1	1

$\Rightarrow$

t_1	t_2	t_3	t_4	τ_1	τ_2	τ_3	τ_4	T	
1	0	0	0	−0.5	0.5	−0.5	0	0	0.5
0	0	1	0	−0.5	0.5	0.5	0	0	0.5
0	1	0	0	0.5	−0.5	0.5	0	0	0.5
0	0	0	1	0	0	0	1	0	1
0	0	0	−1	0.5	0.5	0.5	0	1	1.5

$\Rightarrow$

t_1	t_2	t_3	t_4	τ_1	τ_2	τ_3	τ_4	T	
1	0	0	0	−0.5	0.5	−0.5	0	0	0.5
0	0	1	0	−0.5	0.5	0.5	0	0	0.5
0	1	0	0	0.5	−0.5	0.5	0	0	0.5
0	0	0	1	0	0	0	1	0	1
0	0	0	0	0.5	0.5	0.5	1	1	2.5

We only list the tableaux of using the simplex method. For more information about linear programming and simplex method, we refer the reader to textbooks (e.g., [29]) for details. The right most column of the last tableau gives the optimal solution where $t_1 = t_2 = t_3 = 0.5,\ t_4 = 1$, and $T = 2.5$.

The exhaustive search for all irreducible set covers is computation intensive, and the value of K_{all} might be very large, which again increases the computation of the LP problem. Another approach is to guess a small value of K to solve the small size LP. For example, K can be set to the number of sensors [10]. However, with such a predefined K, the solution of the LP (denote as x_{ik}^* and t_k^*) may be a suboptimal result, and the network lifetime can be further improved. Cardei et al. [10] propose a heuristic which iteratively solves the LP problem to improve the network lifetime.

Iterative LP-based Heuristic (Cardei et al. [10])

Step 1. Solve the LP formulated in (5.4). Let (x_{ik}^*, t_k^*), $i = 1, 2, \ldots, N$, $k = 1, 2, \ldots, K$, be the optimal solution of the LP. Set the network lifetime $T = 0$.

Step 2. The first approximation solution can obtained as follows:

for $k = 1$ to K
  set $x_{ik}^0 = 0$ for all sensors $s_i \in \mathcal{S}$
  set $t_k^0 = \min_j \max_{i \in S(z_j)} x_{ik}^*$
  for $j = 1$ to M
    choose $s_i \in S(z_j)$ such that $x_{ik}^* \geq t_k^0$ and set $x_{ik}^0 = t_k^0$
  endfor
endfor

After the first approximation, each sensor s_i, $i = 1, 2, \ldots, N$, has a remaining lifetime $T_i = 1 - \sum_k x_{ik}^0$, and we set the network lifetime $T = T + \sum_{k=1}^{K} t_k^0$.

Step 3. We iteratively repeat Step 1 and Step 2 by solving the above LP problem with (5.5) substituted by

$$\sum_{k=1}^{K} x_{ik} \leq T_i \quad \text{for all } s_i.$$

The iteration is executed while each target is covered by at least one sensor having the remaining lifetime greater than 0.

Step 4. Return the network lifetime T.

We can also use a modified greedy algorithm to solve the MSC problem. Table 5.4 presents the pseudo-codes for the greedy algorithm proposed in [10]. In the greedy algorithm (called as the CTLW-heuristic), a partition of set covers is constructed in a greedy way (line 04–line 10), where a most beneficial sensor covering a critical target is selected in each stage. A critical target is the most sparsely covered, both in terms of the number of sensors covering it and with regard to the residual energy of these sensors. A sensor has larger benefit if it covers a larger number of

Table 5.4 Pseudo-codes for the CTLW-heuristic [10]

01	Given $\mathcal{S}$, $\mathcal{Z}$, τ, and set lifetime$(s_i) = 1$ for all sensors, and set $k = 1$
02	**while** (all targets can be covered by $\mathcal{S}$)
03	$C_k = \emptyset$; $\mathcal{Z}_{\text{uncvd}} = \mathcal{Z}$; $\mathcal{S}_{\text{left}} = \mathcal{S}$
04	**while** ($\mathcal{Z}_{\text{uncvd}} \neq \emptyset$)
05	find a critical target z^c
07	select a most beneficial sensor $s_i \in \mathcal{S}_{\text{left}}$ covering z^c
08	$C_k = C_k \cup s_i$
09	$\mathcal{Z}_{\text{uncvd}} = \mathcal{Z}_{\text{uncvd}} - S_i$ (S_i is the set of targets covered by s_i)
10	**endwhile**
11	**for** all sensors $s_x \in C_k$
12	lifetime(s_x) = lifetime$(s_x) - \tau$
13	**if** lifetime$(s_x) == 0$, **then** $\mathcal{S} = \mathcal{S} - s_x$ **endif**
16	**endfor**
21	**endwhile**

uncovered targets and if it has more residual energy. After the set cover construction, all sensors in the set cover is set to be active for τ time units, and their lifetime is adjusted accordingly (line 11–line 16). The network lifetime is computed as τK_{greedy}, where K_{greedy} is the number of set covers returned by the greedy algorithm.

Set K-Cover for Minimum Coverage Breach Similar to the Disjoint Set K-Cover problem, the Set K-Cover problem is to construct K set covers, yet without the constraint that these K set covers should be disjoint. Suppose that each sensor has a lifetime of one time unit and consumes the same amount of energy in the active state. In the Disjoint Set K-Cover problem, each sensor can only be included in one set cover, and each set cover is active for one time unit. Hence, in the Disjoint Set K-Cover problem, the network lifetime equals to K. In the nondisjoint Set K-Cover problem, a sensor can be included in more than one set cover, and the active intervals can be different for different set covers. In the nondisjoint Set K-Cover problem, the network lifetime is the sum of these active intervals.

In both the Disjoint Set K-Cover problem and the nondisjoint Set K-Cover problem, a target is allowed to be not covered by any active sensor. Recall that a target is called *breached* if it is not covered by any active sensor. The objectives of the Set K-Cover are to minimize the coverage breach while maximizing the network lifetime. However, they are two conflicting objectives in most cases. We need to balance the two objectives. For example, we can set a threshold as the acceptable minimum network lifetime and then minimize the coverage breach. On the other hand, we can set a threshold as the acceptable maximum allowable coverage breach and then maximize the network lifetime.

The solution to the Set K-Cover problem is a schedule (C_k, t_k), $k = 1, 2, \ldots, K$, and the network lifetime is given by $T \equiv \sum_{k=1} t_k$. Again, we use $\{C_k\}$ to denote the set of targets covered by the set cover C_k. The *total coverage breach* is defined as $\sum_{k=1}^{K} t_k \times (|\mathcal{Z}| - |\{C_k\}|)$, and the *average coverage breach rate* is defined as $\sum_{k=1}^{K} t_k \times (|\mathcal{Z}| - |\{C_k\}|)/\sum_{k=1}^{K} t_k$. Recall that the average coverage breach in the Disjoint Set K-Cover problem is defined as $\sum_{k=1}^{K}(|\mathcal{Z}| - |\{C_k\}|)/K$ (as $t_k = 1$ for all C_k). We can also enforce the bandwidth constraint to the nondisjoint Set K-Cover problem by requiring $|C_k| \leq W$ for all set covers. The nondisjoint Set K-Cover problem can be defined in different ways. We present an example of the Set K-Cover for Minimum Breach (with the minimum lifetime constraint) as follows.

Definition 5.4 (Set K-Cover for Minimum Breach) Given a collection $\mathcal{C}$ of subsets of a finite set $\mathcal{Z}$, a positive integer K, and the network lifetime threshold $T_{th} > 0$, construct a schedule (C_k, t_k) of K set covers $C_1, C_2, \ldots, C_K$ with time intervals $t_1, t_2, \ldots, t_K$ to minimize the coverage breach $\sum_{k=1}^{K} t_k \times (|\mathcal{Z}| - |\{C_k\}|)$. The schedule should also satisfy (1) the bandwidth constraint, $|C_k| \leq W$ for all C_k; and (2) the energy constraint, $\sum_{k=1}^{K} t_k \delta(s_i, C_k) \leq 1$ for all s_i and C_k, where $\delta(s_i, C_k) = 1$ if sensor s_i in the set cover C_k and $\delta(s_i, C_k) = 0$ otherwise.

It can be shown that this problem of Set K-Cover for Minimum Breach is NP-complete. This problem can also be formulated as a linear programming problem, and modified greedy algorithm can be applied to solve this problem [36, 37].

5.1.3 Notes and Comments

Localized Algorithms The target coverage lifetime maximization problem is a basic yet important issue in wireless sensor networks. The common approach is to formulate it as an optimization problem (e.g., a linear programming (LP) problem) and then apply some approximation algorithm (e.g., a greedy algorithm) to find approximate solutions. In the previous subsections, we have introduced some example problems and algorithms. Most of these algorithms are assumed to be executed by a central controller, and only the results are distributed to the sensor nodes. Although centralized algorithms can provide near optimal solutions, they may not be robust enough to adapt to dynamic environments. For example, nodes may suddenly die due to heavy rain or floods before they run out of batteries. One remedy is to apply some mechanism for monitoring the network and reporting unexpected changes back to the central controller. The central controller then computes the algorithm again and disseminates a new schedule to all nodes. Another approach is to design distributed and localized algorithms where each node makes its decision by itself.

A localized algorithm normally requires only local information (e.g., the information from one-hop or two-hop neighbors) for decision making. Sometimes, no local information is required. For example, in the randomized algorithm for the Disjoint Set K-Cover problem, a sensor decides its active state only by itself. On the other hand, a sensor may decide its active state based on the decisions of its neighbors. For example, a sensor may decide to be in a sleep state (not active) if it finds that its active neighbors have already covered its own targets [5, 7]. In the algorithm execution, message exchange is needed and can be implemented by local broadcasting: A sensor broadcasts the information about the targets it covers to its neighbors. A sensor which has just decided its state also needs to broadcast its decision to its neighbors.

Localized algorithms can also be used to emulate the execution of centralized algorithms but with divided yet smaller problem instances. For example, a localized greedy algorithm for constructing K disjoint set covers can be as follows [1]. At first, each node exchanges the information of its covered targets with its neighbors. Then each node randomly chooses its decision making time. When the decision timer expires, a node chooses to be included in the set cover such that it can cover the most number of uncovered targets. After that, the node broadcasts its decision to its neighbors. Obviously, due to the lack of global information, the performance of a localized algorithm is in general not better than its centralized version.

Prasad and Dhawan [15, 27] propose interesting localized algorithms to emulate the centralized nondisjoint set cover construction in a localized way. In their algorithms, each sensor first constructs its local set covers C_k^l, $k = 1, 2, \ldots$. A local set cover C_k^l of a sensor can cover all the targets covered by this sensor and needs not to include this sensor. Each sensor then prioritizes these local set covers according to a so-called *lifetime dependency* graph, where the local set covers C_k^l are as vertices, and an edge is drawn between C_k^l and $C_{k'}^l$ if $C_k^l \cap C_{k'}^l \neq \emptyset$. A sensor starts with the C_k^l of the highest priority to decide its active state. If it can decide its state (either

active or sleep), then it is successful with C_k^l. In such a case, all nodes in C_k^l should be active, and the sensor itself can be either active (if $s_i \in C_k^l$) or sleep (if $s_i \notin C_k^l$). If it is not successful with C_k^l (e.g., some other sensors in C_k^l becomes sleep, and C_k^l is not feasible any more), then it transits to the next best local set cover and continues its decision making process until a local set cover gets satisfied.

Other Variants The lifetime maximization problem discussed so far are all based on the simple sensing disk coverage model. In what follows, we list some other problem variants based on different assumptions, objectives, or other coverage models. Again, all these variants do not consider network connectivity issue. The coverage lifetime maximization with guaranteed network connectivity will be discussed in the next section.

In the sensing disk model, it is often assumed that an active sensor can cover all the targets within the sensing disk centered at itself and with radius of the sensing range. In some cases, an active sensor may choose to monitor only one target or a subset of the targets within its sensing disk [4, 21, 22, 41]. Also in the sensing disk coverage model, it is usually assumed that all active sensor nodes have the same energy consumption rate, regardless how many targets they are covering. Sometimes, the energy consumption rate may depend on the number of covered targets [17]. The complete coverage requirement asks that each target should be covered by at least one active sensor. The reliable coverage normally requires that a target should be covered by more than one active sensor [22, 28]; while the differentiated coverage may ask that different targets should be covered by different numbers of active sensors [28]. The target coverage lifetime maximization problem has also been studied under other coverage models, such as the disk coverage model with adjustable sensing ranges [11], the disk coverage model with multiple sensing modalities [2, 30], the directional coverage model [3, 8, 38], and the estimation coverage model [32, 34].

5.2 Maximizing Connected Target Coverage Lifetime

In the previous section, we have introduced the target coverage lifetime maximization problems without considering the network connectivity. Such problems might be applicable to a kind of one-hop network where all sensor nodes send their sensing data directly to the sink (e.g., the aircraft in the previous example scenario). In some other scenarios, the deployed sensor nodes, together with the sink(s), are often modeled as a multiple-hop network where a sensor may need to send its data to the sink via a multi-hop path consisting of other sensor nodes. In such a multi-hop path, a node may not be necessarily sensing active for monitoring any target but only serves as a *relay* to receive data and retransmit the data.

Let us use an example to illustrate the target coverage problem with the connectivity requirement. As shown in Fig. 5.4, there are one sink, four targets, and 13 randomly deployed sensor nodes. The sensors that can cover one or more targets are indicated by their circles—solid circles for active sensors and dashed circles for other

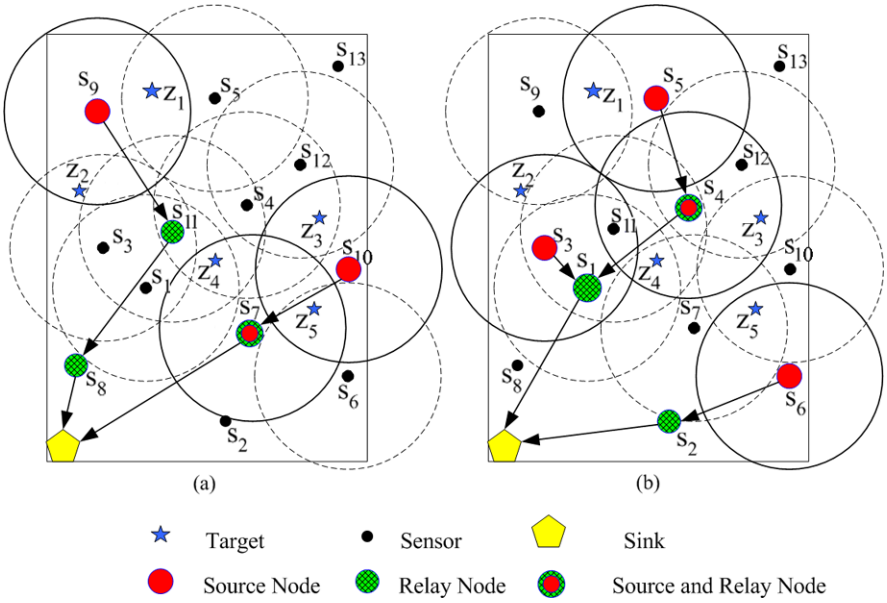

Fig. 5.4 Illustration of the connected target coverage problem: (**a**) one potential solution; (**b**) another potential solution (for the color version, see Color Plates on p. 209)

sensors. Some of nodes may not be able to cover any target, say for example, the sensor nodes s_2, s_8, s_{13} in the figure. However, such sensor nodes may still be useful as they can serve as relays to relay other sensors' data. Arrow lines are used to denote the routes used to relay data from sources to the sink. In Fig. 5.4(a), only three sensors (s_7, s_9, s_{10}) are needed to cover all targets, and the other two sensor nodes (s_8, s_{11}) are used to relay sensing data. A different selection is shown in Fig. 5.4(b), where another subset of sensors is used for covering targets (s_3, s_4, s_5, s_6) and relaying sensing data (s_1, s_2, s_4).

The basic functionalities of a multi-hop WSN for target coverage are to monitor targets (hence generating sensing data) and send the sensing data back to the sink. The above example illustrates that only a subset of the deployed sensors is sufficient to carry out these functionalities, and different subsets can be used in different intervals. In each of such subset, different active sensors can be assigned with different roles. A sensor is called a *source* if it is selected to be active to perform the monitoring task and generates sensing data at a certain rate. A sensor node is called a *relay* if it is selected to be active to relay data only. A relay does not perform the sensing task and does not generate sensing data. An active sensor node can serve as either a source, or a relay, or both. A sensor node that is not active enters into an energy saving *sleep* state.

We use *connected target coverage* problem to refer to the network lifetime maximization problem for target coverage with the connectivity requirement. Two assumptions are made for the network initial configuration.

(1) (Initial connectivity assumption) In the network initialization, all the sensor nodes can reach the sink via either single-hop or multi-hop communications.
(2) (Initial coverage assumption) In the network initialization, each target is covered by at least one sensor.

The *network lifetime* for connected target coverage is defined as the time period from the time when the network is set up to the time that (1) the coverage requirement cannot be satisfied (e.g., one or more targets cannot be covered) or (2) the connectivity requirement cannot be satisfied (e.g., a route cannot be found to send the sensing data to the sink), whichever is earlier. Again, we assume that all sensor nodes have limited lifetime, and hence the network has limited lifetime. A *schedule* is a series of subsets of active sensors (along with the role allocation for each active sensor) and their active intervals. Furthermore, each subset of active sensors should satisfy the coverage and connectivity requirements. The connected target coverage problem is to find the optimal schedule to maximize the network lifetime.

We next elaborate the connected target coverage problem with the complete 1-coverage and 1-connectivity requirements. With such requirements, each target should be covered by at least one active source; and each source can find one route to the sink. In other words, each subset not only serves as a set cover but also forms a routing *tree*. In the graph theory, a tree refers to a graph in which any two vertices are connected by exactly one path. In our context, the routing tree is rooted at the sink, and all sources can find a path to the sink. We call such a subset a *cover tree*. Similar to the maximum set cover problem introduced in the previous section, we can define a maximum cover tree problem with the objective of maximizing the network lifetime and with the coverage and energy constraints plus the connectivity constraint.

Maximum Cover Tree (MCT) Problem for Connected Complete Target Coverage Again, we use $\mathcal{S} = \{s_1, \ldots, s_i, \ldots, s_N\}$ ($|\mathcal{S}| = N$) and $\mathcal{Z} = \{z_1, \ldots, z_j, \ldots, z_M\}$ ($|\mathcal{Z}| = M$) to denote the set of deployed sensor nodes and the set of targets, respectively. We use $\mathcal{R}$ to denote the sink. We assume that all sources have the same data generation rate, i.e., all sources use the same sampling frequency, quantization, modulation, and coding scheme. Therefore, each source generates a fixed amount of bits, denoted by $B(t)$, in a time interval t. In each interval, the state (active/sleep) and the role (source/relay/source + relay) of a sensor remain unchanged within the interval. We further assume that all the generated data in an interval can be sent back to the sink.

The energy consumption model considers three types of energy dissipation, namely, energy consumed for producing, receiving, and sending one data bit. Signaling overheads are assumed to be very small when compared with the sensing data and are not included in the energy consumption model. An energy unit e^s is consumed by a source for producing a bit data. An energy unit energy e^r is consumed by a relay for receiving a bit data, and an energy unit $e^t(d)$ is consumed by a sender (a source or a relay) for sending a bit to a receiver separated with d meters. Here, we consider a distance-dependent transmission energy dissipation $e^t(d) = a + b \times d^c$, where a is the transmitter electronics energy, b is the transmitter amplifier energy,

and c is the radio attenuation factor [18]. For simplicity, we use e^t to represent $e^t(d)$ for a given sender and receiver pair. A source needs to consume $e^s \times B(t)$ energy for its sensing task and at least $e^t \times B(t)$ energy for sending out its sensing data in an active interval t. The energy consumed by a relay depends on the number of bits it will transmit or receive, where the latter further depends on how we construct a connected tree from all the sources to the sink.

We use C_k^s and C_k^r to denote the set of sources and the set of relays in an interval t_k, respectively. The set of active nodes is $C_k = C_k^s \cup C_k^r$, and the set of nodes serving as both source and relay is $C_k^s \cap C_k^r$. We use $\mathcal{T}_k = (C_k^s \cup C_k^r \cup \mathcal{R}, \mathcal{E}_k)$ to denote the constructed routing tree in the interval t_k, where $\mathcal{E}_k$ is the set of edges connecting the active sensors and the sink. The tree $\mathcal{T}_k$ has the following properties: (1) The root of the tree is the sink; (2) Each leaf of the tree is a source sensor; (3) Each target can directly connect to at least one source in the tree (i.e., each target is covered by at least one source). We call $\mathcal{T}_k$ a *cover tree* since it not only covers all targets but also connects every active node to the sink.

In a cover tree $\mathcal{T}_k$, we call a sensor s_i a *descendant* of another sensor $s_{i'}$ if sensor s_i needs $s_{i'}$ to relay its data to the sink, and $s_{i'}$ is called the *ancestor* of s_i. Let $D(s_i, \mathcal{T}_k)$ denote the number of sources among the descendants of sensor s_i in a given cover tree $\mathcal{T}_k$. If a sensor $s_{i'}$ is a leaf, i.e., $s_{i'}$ has no descendants, then $D(s_{i'}, \mathcal{T}_k) = 0$. Obviously, $D(\mathcal{R}, \mathcal{T}_k) = |C_k^s|$. From the above discussion, for a given cover tree $\mathcal{T}_k$ operating in an interval t_k, the energy consumption for a sensor s_i is given by

$$e_k(s_i, \mathcal{T}_k) = \begin{cases} (e^s + e^t)B(t_k), & s_i \in C_k^s \text{ and } s_i \notin C_k^r, \\ (e^r + e^t)B(t_k)D(s_i, \mathcal{T}_k), & s_i \in C_k^r \text{ and } s_i \notin C_k^s, \\ (e^s + e^t)B(t_k) & \\ \quad + (e^r + e^t)B(t_k)D(s_i, \mathcal{T}_k), & s_i \in C_k^s \cap C_k^r, \\ 0, & s_i \notin C_k^s \cup C_k^r. \end{cases} \tag{5.6}$$

A source consumes energy for sensing and sending, and has no descendants. Hence its energy consumption is $e^s B(t_k) + e^t B(t_k)$. A relay consumes energy for receiving and sending, and it has $D(s_i, \mathcal{T}_k)$ descendants. Hence its energy consumption is $(e^r + e^t)B(t_k)D(s_i, \mathcal{T})$. A source plus relay node consumes energy for sensing, receiving, and sending, and hence its energy consumption is $(e^s + e^t)B(t_k) + (e^r + e^t)B(t_k)D(s_i, \mathcal{T}_k)$. A sleep node performs no tasks, and hence its energy consumption is zero.

The above energy consumption model exploits the state and role allocation as well as the tree structure. Although the details of a tree are to be decided, this energy model implicitly guarantees the network connectivity requirement. Furthermore, it has also implicitly included the flow conservation rule via the aid of number of descendants. Again, we call $(\mathcal{T}_k, t_k)$ a *schedule* of cover trees $\mathcal{T}_k$ with their operating intervals t_k, $k = 1, 2, \ldots$. The maximum cover tree problem is to maximize the network lifetime defined by $T \equiv \sum_k t_k$ and is defined as follows.

Definition 5.5 (Maximum Cover Tree (MCT) Problem) Given a set of sensors $\mathcal{S}$, a set of targets $\mathcal{Z}$, and a sink $\mathcal{R}$ satisfying the initial network configuration assump-

tions, find a schedule $(\mathcal{T}_k, t_k)$ of the cover trees $\mathcal{T}_1, \ldots, \mathcal{T}_K$ with operating intervals $t_1, \ldots, t_K$ to maximize $\sum_{k=1}^{K} t_k$. Each cover tree $\mathcal{T}_k = (C_k^s \cup C_k^r \cup \mathcal{R}, \mathcal{E}_k)$ consists of vertices and edges. The vertices of each cover tree consist of the set of sources C_k^s, the set of relays C_k^r, and the sink $\mathcal{R}$; and the edges $\mathcal{E}_k$ specify the routes from sources to the sink. Furthermore, each cover tree can cover all targets ($\{C_k^s\} = \mathcal{Z}$ for all C_k^s), and each sensor does not consume more than its initial energy ($\sum_{k=1}^{K} e_k(s_i, \mathcal{T}_k) \leq E_i$ for all s_i; E_i is the initial energy of s_i).

The optimization version of the MCT problem is formulated as follows:

$$\begin{aligned}
&\text{Maximize:} && T \equiv \sum_{k=1}^{K} t_k \\
&\text{Subject to:} && \sum_{k=1}^{K} e_k(s_i, \mathcal{T}_k) \leq E_i \quad \text{for all } s_i \quad \text{(energy constraint)} \\
& && \mathcal{T}_k = \left(C_k^s \cup C_k^r \cup \mathcal{R}, \mathcal{E}_k\right) \\
& && \mathcal{T}_k \text{ is a tree for all } \mathcal{T}_k \quad \text{(connectivity constraint)} \\
& && \left\{C_k^s\right\} = \mathcal{Z} \quad \text{for all } C_k^s \quad \text{(coverage constraint)}
\end{aligned}$$

In this definition, the number of intervals is denoted by K. Given a finite initial energy and a minimum time interval, the value of K is finite but unknown. In the MCT problem, the sets of active sensors are not required to be disjoint in different intervals, and a sensor can be included in different cover trees as long as its energy constraint is still satisfied. The decision version of the MCT problem is as follows: Given $\mathcal{S}$, $\mathcal{Z}$, and $\mathcal{R}$ satisfying the network initial assumptions, and given a positive number T, find a family of cover trees $\mathcal{T}_1, \ldots, \mathcal{T}_K$ with time intervals $t_1, \ldots, t_K$ such that $\sum_{k=1}^{K} \geq T$. The constraints of energy, connectivity, and coverage are the same as in the optimization version. The MCT problem is NP-complete [45]. It can be easily shown that the MSC problem (Definition 5.3) is a simplified MCT problem without the connectivity constraint. Since the MSC problem is NP-complete, according to the reduction method [16], the MCT problem is also NP-complete. In what follows, we provide another NP-Complete proof for the MCT problem.

Theorem 5.3 (Zhao and Gurusamy [45], Theorem 1) *The MCT problem is NP-complete.*

Proof Given T and an arbitrary family of cover trees $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_K$ with their operating intervals $t_1, t_2, \ldots, t_K$, we can verify in polynomial time whether (1) $\sum_{k=1}^{K} t_k \geq T$, (2) all the targets are covered in each cover tree, and (3) the energy consumption of each sensor node s_i over all the cover trees does not exceed its initial energy E_i.

Therefore, MCP $\in$ NP.

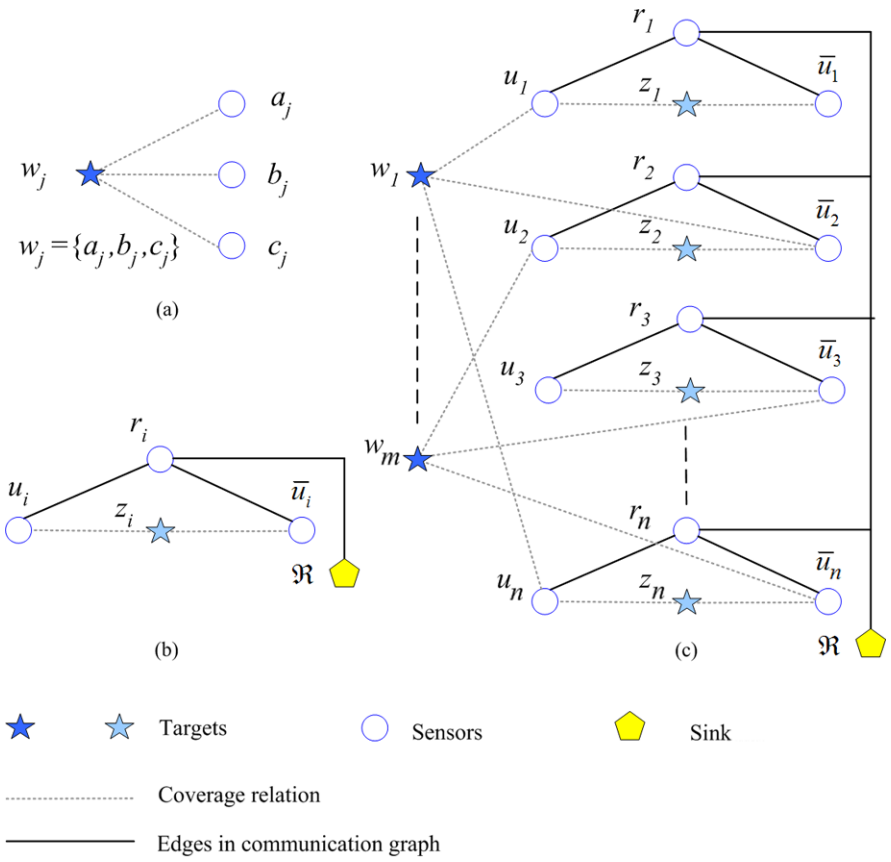

Fig. 5.5 Reduction of 3SAT to MCT problem (reproduced from [45], ©2008, IEEE)

To prove that the MCT problem is NP-hard, we reduce the 3SAT problem to the MCT problem in polynomial time. Let $U = \{u_1, u_2, \ldots, u_n\}$ be the set of variables, and $W = \{w_1, w_2, \ldots, w_m\}$ be the set of clauses in an arbitrary instance of 3SAT.

First, we add a sink node $\mathcal{R}$ into the network. For each variable $u_i \in U$, there is a component (illustrated in Fig. 5.5(b)) composed of one target z_i, three sensor nodes $\mathcal{S}_i = \{u_i, \bar{u}_i, r_i\}$, three communication links $\mathcal{E}_i^c = \{(u_i, r_i), (\bar{u}_i, r_i), (r_i, \mathcal{R})\}$, and two observation links $\mathcal{E}_i^o = \{(u_i, z_i), (\bar{u}_i, z_i)\}$. For each clause $w_j \in W$ with the three literals $a_j, b_j, c_j \in U \cup \bar{U}$, there is a component (illustrated in Fig. 5.5(a)) composed of a target w_j, three sensor nodes a_j, b_j, c_j, and three observation links $\mathcal{E}_j^o = \{(a_j, w_j), (b_j, w_j), (c_j, w_j)\}$.

The construction of our instance of the MCT problem is completed by setting $T = 1$, with the set of sensor nodes $\mathcal{S} = \bigcup_i \mathcal{S}_i$, targets $\mathcal{Z} = (\bigcup_i z_i) \cup (\bigcup_j w_j)$, communication links $\mathcal{E}^c = \bigcup_i \mathcal{E}_i^c$, and observation links $\mathcal{E}^o = (\bigcup_i \mathcal{E}_i^o) \cup (\bigcup_j \mathcal{E}_j^o)$ (illustrated in Fig. 5.5(c)). As an example, in Fig. 5.5(c), the clause w_i is assumed to be $w_1 = \{u_1 \vee \bar{u}_2 \vee u_n\}$. In that case, $a_1 = u_1$, $b_1 = \bar{u}_2$, and $c_1 = u_n$. For each

sensor node, the initial energy E_i is 1 unit, $e^s = e^r$, and $(e^r + e^t)B(1) = 1$. It is easy to see that the construction can be accomplished in polynomial time.

First, we show that a solution for the 3SAT problem can be transformed to the solution for the MCT problem in polynomial time. Suppose that $f : U \rightarrow \{True, False\}$ is a satisfying truth assignment for W. If $f(u_i) = True$, we assign sensor u_i to be the source sensor; otherwise assign $\bar{u}_i$ to be the source sensor. Node r_i is the relay node to connect u_i or $\bar{u}_i$ to the sink node. It is easy to verify that all the targets are covered, the lifetime is 1, and this can be done in polynomial time.

Now, we show that a solution for the MCT problem can be transformed to a solution for the 3SAT problem. Suppose that the schedule of the cover trees $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_K$ and their operating intervals $t_1, t_2, \ldots, t_K$ is a feasible solution for the MCT problem and that $\sum_{k=1}^{K} t_k \geq 1$. As target z_j can be covered by sensor u_i and $\bar{u}_i$ only, at least one of them should be active as the source node at any time, and thus r_i must be active as the relay node all the time. Further, u_i and $\bar{u}_i$ cannot be simultaneously chosen as the source nodes; otherwise r_i will consume more than 1 unit of energy if the lifetime is 1. Thus in any cover tree $\mathcal{T}_k$, one and only one sensor among sensors u_i and $\bar{u}_i$ (but not both) should be chosen as the source node. Furthermore, if target z_j is covered by a source node $a_j \in U \cup \bar{U}$, the corresponding clause w_j must be true if we set $a_j = 1$. Therefore, assigning the corresponding literals of the source nodes in any cover tree $\mathcal{T}_k$, $1 \leq k \leq K$, to be true gives a satisfying true assignment for W. Therefore, the MCT problem is NP-hard.

Since the MCT problem belongs to class NP and is NP-hard, it is NP-complete. □

Since the MCT problem is NP-complete, we resort to approximation algorithms to solve the problem. Again, we can use a modified greedy algorithm to solve the MCT problem. Table 5.5 presents the pseudo-codes for the greedy algorithm proposed in [42, 45]. The heuristic algorithm uses a greedy method to select the sources to cover all the targets, and it couples the communication cost and sources selection. It is called *Communication Weighted Greedy Cover* (CWGC). The inputs of the algorithm are $\mathcal{S}$, $\mathcal{Z}$, $\mathcal{R}$, and E_i. The output of the algorithm is a sequence of cover trees $\mathcal{T}_1, \ldots, \mathcal{T}_K$ and their operating intervals $t_1, \ldots, t_K$. The number of the cover trees in the sequence is denoted as K. In the table, the following notation is used:

$\mathcal{S}_l$	set of live sensors;
$\mathcal{S}_s$	set of live sensors that can cover targets;
$\mathcal{Z}_s$	set of targets that can be covered by s;
w_s	path weight of sensor s;
$W(s)$	profit of sensor $s \in \mathcal{S}_s$;
$R(s, \mathcal{T})$	route from the sensor s to the sink $\mathcal{R}$ in tree $\mathcal{T}$, $R(s, \mathcal{T}) \equiv (s, s_1, \ldots, \mathcal{R})$;
$\overline{R}(s, \mathcal{R})$	set of sensors in route $R(s, \mathcal{T})$ excluding sensor s and the sink $\mathcal{R}$

Table 5.5 Pseudo-codes for the CWGC algorithm [45]

```
01  S_l = S; S_s = ∅; k = 1
02  for each s ∈ S_l
03    E_s^r = E_s
04    if Z_s ≠ ∅, S_s = S_s ∪ {s}; endif
05  endfor
06  while ⋃_{s∈S_s} Z_s = Z and S_l ≠ ∅
07  phase 1:
08    for each link (s_i, s_i'), w_ii' = e_ii'^t × E_i/E_i^r; endfor
09    Build a MWCT T_m connecting each sensor s ∈ S_l to the sink
10  phase 2:
11    S_s' = ∅; Z' = ∅; T_k = ∅; t_k = τ;
12    while Z' ≠ Z,
13      Find a sensor s* ∈ S_s − S_s' with the maximum profit W(s*)
14      S_s' = S_s' ∪ {s*}; Z' = Z' ∪ Z_s*;
15      for each s ∈ R̄(s*, T_k),
16        w_s = w_s + (e^t + e^r)B(t_k) × w_s/E_s^r;
17      endfor
18    endwhile
19  phase 3:
20    for each s ∈ S_s', T_k = T_k ⊎ R(s, T_m); endfor
21    for each s ∈ T_k, t_k = min(t_k, t_k × E_s^r/e(s, T_k)); endfor
22    for each s ∈ T_k, E_s^r = E_s^r − e(s, T_k); endfor
23    Remove dead and isolated nodes; k = k + 1;
24  endwhile
```

The algorithm is initialized before constructing cover trees (lines 1–5). The algorithm repeatedly builds cover trees and stops until no new cover tree can be build (i.e., the network lifetime is reached). Each cover tree operates for a fixed time interval τ, unless some sensors in the cover tree will die before the end of the time interval due to the lack of energy. In that case, the operating time of the cover tree is determined by the sensor which has the least operating time until death, i.e., $\min_{s_i \in \mathcal{T}_k}(\tau \times E_i^r / e_k(s_i, \mathcal{T}_k))$. Therefore, the active time of a cover tree $\mathcal{T}_k$ is given by

$$t_k = \min\left(\tau, \min_{s_u \in \mathcal{T}_k}\left(\tau \times \frac{E_i^r}{e_k(s_i, \mathcal{T}_k)}\right)\right), \tag{5.7}$$

where E_i^r is the residual energy of sensor s_i at the beginning of operating cover tree $\mathcal{T}_k$.

If a sensor has no residual energy, we call it a *dead* sensor. If a sensor has residual energy but cannot find a route from itself to the sink without traversing a dead sensor, we call it an *isolated* sensor. Before each iteration of building a new cover tree, the

dead and isolated sensors in the network will be removed. The graph used to build the new cover tree contains only the live sensors.

In each iteration, the algorithm works in three phases to construct a cover tree. In the first phase (phase 1), an energy-aware communication tree is constructed connecting all the live sensor nodes to the sink. The algorithm then greedily selects source sensors that can cover all the targets (phase 2), considering both the number of uncovered targets in the sensing area and the possible communication cost from sensors to the sink. Finally, in phase 3, the new cove tree is constructed based on the communication tree built in phase 1 and the source sensor set selected in phase 2.

In phase 1, we first assign a weight $w_{ii'}$ to each link between live sensors s_i and $s_{i'}$ (line 8) which reflects both the communication energy consumption on the link and the residual energy level of the sender. For example,

$$w_{ii'} = e^t_{ii'} \times E_i / E^r_i .$$

A *minimum weight communication tree* (MWCT) is then constructed connecting all sensors such that the sum of the link weights from each node to the sink is minimized (line 9). Known techniques to find the shortest path tree (e.g., Dijkstra's Algorithm) can be used to construct the tree.

In phase 2, we use a greedy method to choose the sources until all the targets are covered. The greedy method first assigns each sensor s a profit value $W(s)$ and then repeatedly chooses the sensor with the highest $W(s)$ into the source set (lines 13–17). The profit function is given by

$$W(s) \equiv \frac{|\mathcal{Z}_s - \mathcal{Z}_s \cap \mathcal{Z}'|}{w_s},$$

where $|\mathcal{Z}_s - \mathcal{Z}_s \cap \mathcal{Z}'|$ is the number of uncovered targets that can be covered by s, and w_s is the path weight of node s (the sum of link weights in the route $R(s, \mathcal{T})$. After each new source is selected, the path weights of the upstream nodes in MWCT are updated (line 16).

In phase 3, the cover tree is extracted as a subtree of the MWCT based on the selected sources (line 20). After the cover tree is built, we use the energy consumption model, (5.6), for each sensor in the cover tree. The operation duration for the cover tree is then calculated using (5.7) (line 21). Finally, the residual energy for sensors in the cover tree is updated according to their functionalities, and dead and isolated sensors are removed (line 23).

5.2.1 Notes and Comments

The lifetime maximization problem discussed in the previous section considers the basic 1-coverage and 1-connectivity requirements. A generalization of the connected target coverage problem is to enforce k_1-coverage and k_2-connectivity requirements such that every target is covered by k_1 different active sensors, and such

active sensors can form a k_2-connected network [20, 39, 40]. A k_2-connected network indicates that for any two nodes in the network, there are k_2 node-disjoint routes from one node to another. This general lifetime maximization problem with k_1-coverage and k_2-connectivity requirements can also be formulated as a linear programming problem, and modified greedy algorithms can be applied to provide approximation solutions.

Some other variants of the lifetime maximization problem for connected target coverage have also been investigated in literature. For example, only disjoint cover trees are allowed to be constructed [19]. Some consider the scenario that each active sensor can only cover one or a subset of targets within its sensing range [23, 24, 43, 44]. In the sensing disk model, each sensor may have different sensing ranges, and the sensing range is also needed to be scheduled [25]. Some other sensing models such as estimation coverage [33] has also been considered for connected target coverage [35].

References

1. Abrams, Z., Goel, A., Plotkin, S.: Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In: ACM International Symposium on Information Processing in Sensor Networks (IPSN), pp. 424–432 (2004)
2. Ahuja, S.K., Kini, S., Ramasubramanian, S.: Bounds on coverage time and node density for multi-modality sensing. Ad Hoc Networks (Elsevier) (2009). doi:10.1016/j.adhoc.2009.01.006
3. Ai, J., Abouzeid, A.A.: Coverage by directional sensors in randomly deployed wireless sensor networks. Journal of Combinatorial Optimization **11**(1), 21–41 (2006)
4. Aly, M., Pruhs, K., Znati, T., Hunsaker, B.: The coverage problem for myopic sensors. In: IEEE International Conference on Wireless Networks, Communications and Mobile Computing, pp. 964–968 (2005)
5. Berman, P., Calinescu, G., Shah, C., Zelikovsky, A.: Power efficient monitoring management in sensor networks. In: IEEE Wireless Communications and Networking Conference (WCNC), pp. 2329–2334 (2004)
6. Berman, P., Calinescu, G., Shah, C., Zelikovsky, A.: Efficient energy management in sensor networks. In: Pan, Y., Xiao, Y. (eds.) Ad Hoc and Sensor Networks: Wireless Networks and Mobile Computing. Nova Science, New York (2006)
7. Brinza, D., Zelikovsky, A.: Deeps: Deterministic energy-efficient protocol for sensor networks. In: IEEE the 7th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD), pp. 261–266 (2006)
8. Cai, Y., Lou, W., Li, M., Li, X.Y.: Target-oriented scheduling in directional sensor networks. In: IEEE Infocom, pp. 1–9 (2007)
9. Cardei, M., Du, D.Z.: Improving wireless sensor network lifetime through power aware organization. Wireless Networks **11**(3), 333–340 (2005)
10. Cardei, M., Thai, M.T., Li, Y., Wu, W.: Energy-efficient target coverage in wireless sensor networks. In: IEEE Infocom, pp. 1976–1984 (2005)
11. Cardei, M., Wu, J., Lu, M., Pervaiz, M.O.: Maximum network lifetime in wireless sensor networks with adjustable sensing ranges. In: IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 438–445 (2005)
12. Cheng, M.X., Ruan, L., Wu, W.: Coverage breach problems in bandwidth-constrained sensor networks. ACM Transactions on Sensor Network (ToSN) **3**(2), 1–23 (2007)

13. Deshpande, A., Khuller, S., Malekian, A., Toossi, M.: Energy efficient monitoring in sensor networks. In: The 8th Latin American Symposium on Theoretical Informatics (also in LNCS, vol. 4957), pp. 436–448 (2008)
14. Deshpande, A., Khuller, S., Malekian, A., Toossi, M.: Energy efficient monitoring in sensor networks. Technical Report, Computer Science Department, University of Maryland, USA (2009)
15. Dhawan, A., Prasad, S.K.: Energy efficient distributed algorithm for sensor target coverage based on properties of an optimal schedule. In: The 15th International Conference on High Performance Computing (HiPC), also in LNCS, vol. 5374, pp. 269–281 (2008)
16. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
17. Gu, Y., Ji, Y., Li, J., Zhao, B.: Qos-aware target coverage in wireless sensor networks. Wireless Communications and Mobile Computing (Wiley) (2009). doi:10.1002/wcm.748
18. Heinzelman, W.B., Chandrakasan, A.P., Balakrishnan, H.: An application-specific protocol architecture for wireless microsensor networks. IEEE Transactions on Wireless Communications **1**(4), 660–670 (2002)
19. Jaggi, N., Abouzeid, A.A.: Energy-efficient connected coverage in wireless sensor networks. In: The 4th Asian International Mobile Computing Conference (AMOC), pp. 77–86 (2006)
20. Li, D., Cao, J., Liu, M., Zheng, Y.: K-connected target coverage in wireless sensor networks. In: The 1st International Conference on Combinatorial Optimization and Applications (COCOA), also in LNCS, vol. 4616, pp. 20–31 (2007)
21. Liu, H., Wan, P., Yi, C.W., Jia, X., Kakki, S., Pissinou, N.: Maximal lifetime scheduling in sensor surveillance networks. In: IEEE Infocom, pp. 2482–2491 (2005)
22. Liu, H., Wan, P., Jia, X.: Maximal lifetime scheduling for k to 1 sensor-target surveillance networks. Computer Networks (Elsevier) **50**(2), 2839–2854 (2006)
23. Liu, H., Wan, P., Jia, X.: Maximal lifetime scheduling for sensor surveillance systems with k sensors to one target. IEEE Transactions on Parallel and Distributed Systems **17**(12), 1–11 (2006)
24. Liu, H., Jia, X., Wan, P.J., Yi, C.W., Makki, S.K., Pissinou, N.: Maximizing lifetime of sensor surveillance systems. IEEE/ACM Transactions on Networking **15**(2), 334–345 (2007)
25. Lu, M., Wu, J., Cardei, M., Li, M.: Energy-efficient connected coverage of discrete targets in wireless sensor networks. International Journal of Ad Hoc and Ubiquitous Computing (Interscience) **4**(3–4), 137–147 (2009)
26. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, Cambridge (1995)
27. Prasad, S.K., Dhawan, A.: Distributed algorithms for lifetime of wireless sensor networks based on dependencies among cover sets. In: The 14th International Conference on High Performance Computing (HiPC), also in LNCS, vol. 4973, pp. 381–392 (2007)
28. Pyun, S.Y., Cho, D.H.: Power-saving scheduling for multiple-target coverage in wireless sensor networks. IEEE Communications Letters **13**(2), 130–132 (2009)
29. Rolf, H.L.: Finite Mathematics, 7th Edition. Engage Learning (2007)
30. Shih, K.P., Chen, H.C., Chou, C.M., Liu, B.J.: On target coverage in wireless heterogeneous sensor networks with multiple sensing units. Journal of Network and Computer Applications (2009). doi:10.1016/j.jnca.2009.01.002
31. Slijepcevic, S., Potkonjak, M.: Power efficient organization of wireless sensor networks. In: IEEE International Conference on Communications (ICC), vol. 2, pp. 472–476 (2001)
32. Vashistha, S., Azad, A.P., Chockalingam, A.: Efficient scheduling of sensor activity for information coverage in wireless sensor networks. In: IEEE the 2nd International Conference on Communication System Software and Middleware (COMSWARE) (2007)
33. Wang, B., Wang, W., Srinivasan, V., Chua, K.C.: Information coverage for wireless sensor networks. IEEE Communications Letters **9**(11), 967–969 (2005)
34. Wang, B., Chua, K.C., Srinivasan, V., Wang, W.: Scheduling sensor activity for point information coverage in wireless sensor networks. In: International Symposium on Modelling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt) (2006)

35. Wang, B., Vikram, S., Chua, K.C., Wang, W.: Information coverage and network lifetime in energy constrained wireless sensor networks. In: IEEE the 32rd Conference on Local Computer Networks (LCN), pp. 512–519 (2007)
36. Wang, C., Thai, M.T., Li, Y., Wang, F., Wu, W.: Minimum coverage breach and maximum network lifetime in wireless sensor networks. In: IEEE Global Telecommunications Conference (Globecom), pp. 1–6 (2007)
37. Wang, C., Thai, M.T., Li, Y., Wang, F., Wu, W.: Optimization scheme for sensor coverage scheduling with bandwidth constraints. Optimization Letters **3**(1), 63–75 (2009)
38. Wang, J., Niu, C., Shen, R.: Randomized approach for target coverage scheduling in directional sensor network. In: International Conference on Embedded Software and Systems (ICESS), also in LNCS, vol. 4527, pp. 379–390 (2007)
39. Yang, S., Dai, F., Cardei, M., Wu, J.: On multiple point coverage in wireless sensor networks. In: IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), pp. 757–764 (2005)
40. Yang, S., Dai, F., Cardei, M., Wu, J., Patterson, F.: On connected multiple point coverage in wireless sensor networks. International Journal of Wireless Information Networks (Springer) **13**(4), 289–301 (2006)
41. Zhao, Q., Gurusamy, M.: Lifetime maximization using observation time scheduling in multihop sensor networks. In: IEEE/CreateNet International Workshop on Broadband Advanced Sensor Networks (BroadNets) (2005)
42. Zhao, Q., Gurusamy, M.: Maximizing network lifetime for connected target coverage in wireless sensor networks. In: IEEE International Conference on Wireless and Mobile, Networking and Communications (WiMob), pp. 94–101 (2006)
43. Zhao, Q., Gurusamy, M.: Optimal observation scheduling for connected target coverage problem in wireless sensor networks. In: IEEE International Conference on Communications (ICC), pp. 3728–3733 (2007)
44. Zhao, Q., Gurusamy, M.: Connected k-target coverage problem in wireless sensor networks with different observation scenarios. Computer Networks (Elsevier) **52**(11), 2205–2220 (2008)
45. Zhao, Q., Gurusamy, M.: Lifetime maximization for connected target coverage in wireless sensor networks. IEEE/ACM Transactions on Networking **16**(6), 1378–1391 (2008)

Part III
Area Coverage Problems

Chapter 6
Critical Sensor Density

Abstract Before network deployment, one may want to know how many sensor nodes are needed such that every point of the sensor field can be covered by at least one sensor. *Sensor density* is defined as the number of nodes per unit area. In a homogeneous sensor network, the *critical sensor density* (CSD) provides an insight on the minimal number of nodes required for complete area coverage. In deterministic sensor placement, it is desirable to know not only the minimal number of nodes but also their locations. For a homogeneous network, a basic *placement pattern* can be repeated to tile-up the whole sensor field. A placement pattern is optimal if it requires the minimal number of nodes to completely cover the sensor field, and an optimal placement pattern determines the CSD for deterministic sensor placements. In random sensor deployments, the analysis for CSD often applies an asymptotic approach, but its result provides a lower bound of CSD for a sensor field with finite area: The number of nodes should be no smaller than the area of the sensor field times the critical sensor density to ensure complete area coverage (almost surely) in each deployment. In particular, the following questions are addressed in this chapter:

What is the optimal placement pattern in deterministic node deployments and how to derive the critical sensor density for random node deployments?

We first introduce the optimal placement pattern for deterministic node placement and then review existing approaches for CSD analysis in random network deployment.

6.1 Deterministic Node Placement

6.1.1 Node Placement in Two-Dimensional Field

In deterministic node placements, a basic *placement pattern* consisting of only a few nodes can be repeated to tile-up the whole sensor field. A regular tiling of polygons in two dimensions (or polyhedra in three dimensions) is also called a *tessellation*. A placement pattern can be a polygon, and sensor nodes are put at the polygon's vertices. If a polygon is completely covered by the sensors at its vertices, then the 2D

B. Wang, *Coverage Control in Sensor Networks,*
Computer Communications and Networks,
DOI 10.1007/978-1-84996-059-5_6,

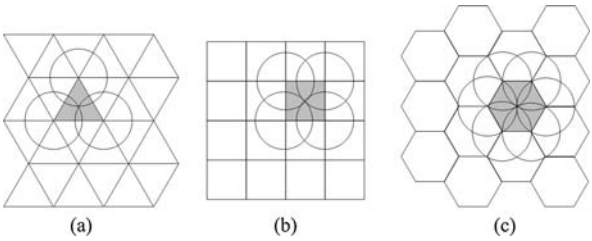

Fig. 6.1 Tessellation of using (**a**) regular triangles, (**b**) regular squares, and (**c**) regular hexagons

plane can be completely covered by the tessellation of such polygons. For example, Fig. 6.1 illustrates three canonical tessellations by regular triangles, regular squares, and regular hexagons.

The critical sensor density λ_{csd} for a tessellation can be computed as follows. Let A_p denote the area of a placement pattern (a polygon), N_p the number of nodes that compose a pattern, and N_n the number of pattern blocks that share a node. Then λ_{csd} can be computed as

$$\lambda_{\text{csd}} = \frac{N_p}{A_p N_n}. \tag{6.1}$$

Let R_s denote the radius of the sensing disk. Then we can compute the CSD for regular triangular tessellation λ^t_{csd}, regular square tessellation λ^s_{csd}, and regular hexagonal tessellation λ^h_{csd} as follows:

$$\lambda^t_{\text{csd}} = \frac{3}{\frac{3\sqrt{3}}{4} R_s^2 \times 6} = \frac{2\sqrt{3}}{9} \times \frac{1}{R_s^2}, \tag{6.2}$$

$$\lambda^s_{\text{csd}} = \frac{4}{2R_s^2 \times 4} = \frac{1}{2} \times \frac{1}{R_s^2}, \tag{6.3}$$

$$\lambda^h_{\text{csd}} = \frac{6}{\frac{3\sqrt{3}}{2} R_s^2 \times 3} = \frac{4\sqrt{3}}{9} \times \frac{1}{R_s^2}. \tag{6.4}$$

Obviously, $\lambda^t_{\text{csd}} < \lambda^s_{\text{csd}} < \lambda^h_{\text{csd}}$, and the regular triangular tessellation has the minimum density requirement among the three tessellations. Actually, regular triangular tessellation is the optimal tessellation in terms of the minimum number of sensing disks required for complete area coverage. This can be deduced from the following theorem proved by Kershner [14] in 1939.

Theorem 6.1 (Kershner [14]) *Let A denote the volume of a bounded plane point set $\mathcal{A}$, and let $N(r)$ be the minimum number of disks of radius r which can completely cover $\mathcal{A}$. Then*

$$\lim_{r \to 0} \frac{\pi r^2 N(r)}{A} = \frac{2\sqrt{3}\pi}{9}. \tag{6.5}$$

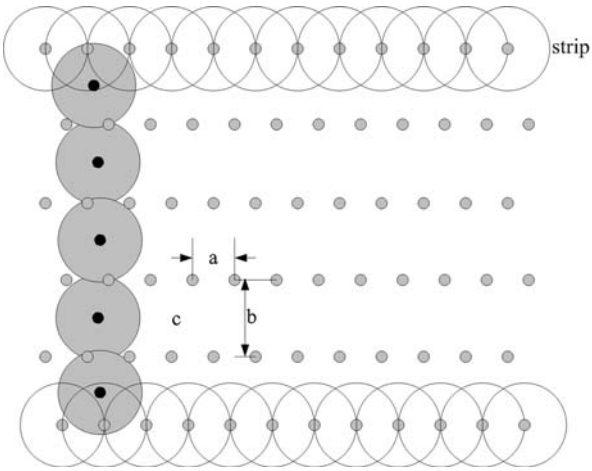

Fig. 6.2 A strip-based placement pattern for complete coverage and 1-connectivity, $a = \min\{R_c, \sqrt{3}R_s\}$, $b = R_s + \sqrt{R_s^2 - a^2/4}$. When $R_c < \sqrt{3}R_s$, one vertical strip is added to ensure 1-connectivity

For a polygon $\mathcal{A}$, A actually is its area. This theorem is a general statement which does not involve with any restriction on the nature of permissible coverings or on the nature of the given region. The left side of (6.5) is the ratio between the total area of the disks covering $\mathcal{A}$ and the area of $\mathcal{A}$. The constant $\frac{2\sqrt{3}\pi}{9}$ of the right side of (6.5) may be thought of as measuring the proportion of unavoidable overlapping. Now let us examine the regular triangular placement pattern. Suppose that the triangular tessellation is used in a very large field. Then the ratio between the total area of all disks and the area of the whole field can be computed $\frac{\pi 2\sqrt{3}R_s^2}{9R_s^2} = \frac{2\sqrt{3}\pi}{9}$. This indicates that the regular triangular tessellation achieves the minimum overlapping. Hence it requires the minimum number of nodes and is an optimal tessellation.

In order to form a connected network, the transmission range R_c should satisfy $R_c \geq \sqrt{3}R_s$ for the regular triangle tessellation, $R_c \geq \sqrt{2}R_s$ for the regular square tessellation, and $R_c \geq R_s$ for the regular hexagon tessellation. Obviously, network connectivity poses another restriction on sensor density. The regular triangle tessellation is also optimal for network connectivity if and only if $R_c \geq \sqrt{3}R_s$. However, it is not uncommon that R_c may be smaller than $\sqrt{3}R_s$. Some researchers [5, 11, 12, 21, 28] have proposed and analyzed a kind of strip-based node placement pattern to address the network connectivity for $R_c < \sqrt{3}R_s$. As shown in Fig. 6.2, a horizonal strip of nodes is formed by putting together nodes at regular separation of $a = \min\{R_c, \sqrt{3}R_s\}$. These strips are deployed horizontally with alternate rows shifted to the right by a distance of $a/2$. The vertical separation between the neighboring strips is $b = R_s + \sqrt{R_s^2 - a^2/4}$. When $R_c > \sqrt{3}R_s$, then $a = \sqrt{3}R_s$ and $b = \frac{3}{2}R_s$, and the deployment pattern is actually a regular triangular tessellation. When $R_c < \sqrt{3}R_s$, the nodes within a strip are connected, but these horizonal strips are not connected. To form a connected network for all nodes, some additional nodes are placed as a vertical strip to connect these horizonal strips. As shown in Fig. 6.2, one vertical strip is added to ensure 1-connectivity. In fact, k-connectivity can be achieved by adding k such vertical strips. When the number of horizonal

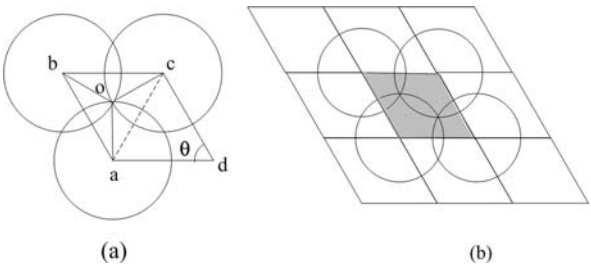

Fig. 6.3 Illustration of rhombus (**a**) and rhombus tessellation (**b**)

strips is much larger than the number of vertical strips, the vertical nodes contribute little to the critical sensor density. Bai et al. [5] prove that the strip-based placement is asymptotically optimal for achieving complete coverage and 1-connectivity even when $R_c < \sqrt{3}R_s$.

For different values of $\frac{R_c}{R_s}$, it is interesting to compare the density requirements of several regular tessellations. Besides the aforementioned regular triangle, square, and hexagon tessellation, rhombus is also often used for tessellation. As shown in Fig. 6.3, the four edges of the rhombus are of the same length, and the acute inner angle is denoted by θ. The triangle $\triangle abc$ is completely covered by the three sensors located at the vertices of $\triangle abc$ iff the circumcenter of the triangle $\triangle abc$ is covered by the three sensors. Let $\eta = \frac{A_p N_n}{N_p}$ denote the *area per node*, where A_p, N_n, and N_p are defined in (6.1). Obviously, the maximum $\eta_{\max}$ is achieved at the critical sensor density λ_{csd}. For the general values of R_s and R_c, the maximum area per node for the regular triangle, square, and hexagon tessellation, denoted by $\eta^t_{\max}$, $\eta^s_{\max}$, and $\eta^h_{\max}$, respectively, are given by

$$\eta^t_{\max} = \frac{3}{2}\sqrt{3}\left(\min\left\{R_s, \frac{\sqrt{3}}{3}R_c\right\}\right)^2, \tag{6.6}$$

$$\eta^s_{\max} = 2\left(\min\left\{R_s, \frac{\sqrt{2}}{2}R_c\right\}\right)^2, \tag{6.7}$$

$$\eta^h_{\max} = \frac{3}{4}\sqrt{3}\big(\min\{R_c, R_s\}\big)^2. \tag{6.8}$$

For the rhombus tessellation, Bai et al. [5] derive the maximum area per node ($\eta^r_{\max}$) by the following theorem.

Theorem 6.2 (Bai et al. [5], Theorem 5.1) *Let $\eta^r(R_s, R_c, \theta)$ denote the area per node in a rhombus tessellation that provides both complete coverage and network connectivity, where R_s is the sensing radius, R_c the transmission radius, and θ the inner acute angle of a rhombus. Then*

$$\eta^r(R_s, R_c, \theta) = f^2(R_s, R_c, \theta)\sin\theta, \tag{6.9}$$

where $f(R_s, R_c, \theta) = \min\{R_c, 2R_s \cos\frac{\theta}{2}\}$. *The maximum value of* $\eta^r(R_s, R_c, \theta)$, $\eta^r_{\max}$, *occurs at* $\theta = \frac{\pi}{2}$ *when* $\frac{R_c}{R_s} \le \sqrt{2}$; *at* $\theta = \pi - 2\arcsin\frac{R_c}{2R_s}$ *when* $\sqrt{2} \le \frac{R_c}{R_s} \le \sqrt{3}$; *and at* $\theta = \frac{\pi}{3}$ *when* $\sqrt{3} \le \frac{R_c}{R_s}$.

Among the aforementioned four tessellations, the following theorem summarizes which placement pattern is better than the other three for different ranges of $\frac{R_c}{R_s}$.

Theorem 6.3 (Bai et al. [5], Theorem 5.1) *Consider a network of homogeneous sensors with sensing radius* R_s *and communication radius* R_c *deployed over a unit square region that is required to provide both complete coverage and network connectivity. Let* $\eta_{\max}$ *be the maximum area per node of the four regular placement patterns*: *hexagon*, *square*, *rhombus*, *and triangle*. *Then*

$$\eta_{\max} = \begin{cases} \eta^h_{\max} & \text{when } 0 < \frac{R_c}{R_s} \le \frac{1}{2} \times 3^{\frac{3}{4}}, \\ \eta^s_{\max} & \text{when } \frac{1}{2} \times 3^{\frac{3}{4}} \le \frac{R_c}{R_s} \le \sqrt{2}, \\ \eta^r_{\max} & \text{when } \sqrt{2} \le \frac{R_c}{R_s} \le \sqrt{3}, \\ \eta^t_{\max} & \text{when } \sqrt{3} \le \frac{R_c}{R_s}, \end{cases} \tag{6.10}$$

where $\eta^h_{\max}$, $\eta^s_{\max}$, $\eta^r_{\max}$, *and* $\eta^t_{\max}$ *are defined in* (6.8), (6.7), (6.9), *and* (6.6), *respectively.*

The corresponding CSD of these regular tessellations can be computed as the inverse of $\eta_{\max}$ for different values of $\frac{R_c}{R_s}$. Bai et al. [6, 7] extend their study of optimal node placement pattern to higher connectivity requirement (up to 6-connectivity).

6.1.2 Node Placement in Three-Dimensional Space

Most wireless terrestrial networks are based on the two-dimensional (2D) scenario where all nodes of a network are assumed to reside on a plane. This 2D assumption can be justified if the size (i.e., the length and width) of a network is much larger than the differences in the third dimension (i.e., the height) of the nodes' locations. The 2D assumption cannot be applied in networks where the difference of the heights of the nodes' locations is comparable to the size of the network. Recently, many three-dimensional (3D) ad hoc and sensor networks have attracted a lot of research attention [2]. For example, in an underwater sensor network for ocean column monitoring, nodes are placed on buoys that are anchored at different depths of the water.

The 3D coverage model of a sensor is often modeled as a *sphere* with radius R_s. Each point within the sphere is covered by this sensor. In 3D deterministic sensor deployments, the critical sensor density corresponds to using the minimal number of spheres to fill in a three-dimensional space so that every point is within the sensing sphere of at least one sensor. Overlapping among spheres is allowed. Actually,

if complete coverage of a 3D space is required, the overlapping among spheres is unavoidable. The coverage problem in 3D networks is much more difficult than that in 2D networks. In 2D, the Kershner's theorem (cf. Theorem 6.1) with an elegant proof gives the asymptotical optimal ratio between the area of a 2D plane and the area of the overlapped disks which completely cover the plane. Unfortunately, a similar result in 3D is still an open problem.

Tessellation by repeating a basic placement pattern is a useful approach to provide the insight on how to optimally place sensors. In 2D, tessellation is applied to polygons, and in 3D, tessellation is applied to polyhedrons. The tessellation polyhedron (also called space-filling polyhedron) fills a volume without any overlap or gap. The objective is to find a space-filling polyhedron that is inscribed within the sensing sphere and best approximates the volume of the sphere. A more general statement of the optimal space filling problem is the *Kelvin's Conjecture*. In 1887, Lord Kelvin asked the following question [13]:

> *"What is the optimal way to fill a three-dimensional space with cells of equal volume so that the surface area* (*interface area*) *is minimized?"*

Kelvin's answer to his question is a 14-sided truncated octahedron having a very slight curvature of the hexagonal faces. However, Kelvin could not prove the optimality of his proposed structure. Indeed, the optimal solution to Kelvin's Conjecture is still not known yet.

The Kelvin's conjecture is equivalent to finding a space-filling polyhedron with the highest *isoperimetric quotient*. Let V and S denote the volume and surface area of a polyhedron, respectively. The isoperimetric quotient of a polyhedron is defined as $\frac{36\pi V^2}{S^3}$. A sphere has the highest isoperimetric quotient of 1, and the Kelvin's 14-sided truncated octahedron has the isoperimetric quotient of 0.753367. In order to find a space-filling polyhedron, Alam and Haas [3] propose a new metric, called *volumetric quotient*, to describe how a space-filling polyhedron inscribed within a sphere approximates the sphere. A space-filling polyhedron is inscribed within a sphere if its center coincides with the sphere center, and the maximum distance from its center to any vertex is equal to the sphere radius. Formally, the volumetric quotient is defined as the ratio between the volume of the polyhedron V, and the volume of its circumsphere with radius R_s, that is, $\frac{V}{\frac{4}{3}\pi R_s^3}$. It is well known that among all polyhedrons, the following claims hold:

(1) For a given volume, a sphere has the smallest surface area.
(2) For a given surface area, a sphere has the largest volume.

Alam and Haas [3] conjecture that the space-filling polyhedron with the minimum ratio of surface area to volume best approximates the sphere. They also compare the volumetric quotient for several common polyhedrons. Figure 6.4 illustrates these polyhedrons: (a) cube, (b) hexagon prism, (c) rhombic dodecahedron, and (d) truncated octahedron.

Let R_s denote the radius of a sensing sphere. For the aforementioned four polyhedrons inscribed within a sensing sphere, the edge length, volume and volumetric quotient are computed as follows.

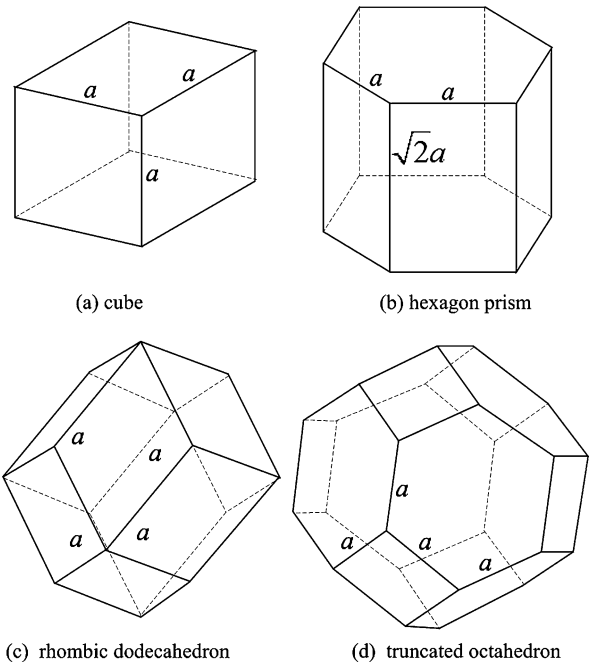

Fig. 6.4 Four space-filling polyhedrons: (**a**) cube, (**b**) hexagonal prism, (**c**) rhombic dodecahedron, and (**d**) truncated octahedron

- A cube has 6 square faces, 12 edges, and 8 vertices, and all the edges have the same length. For a cube inscribed within the sensing sphere, all of its 8 vertices are on the spherical surface. The length of its edge is $\frac{2}{\sqrt{3}}R_s$, and its volume is $V_c = (\frac{2}{\sqrt{3}}R_s)^3 = \frac{8}{3\sqrt{3}}R_s^3$. The volumetric quotient is $\frac{V_c}{\frac{4}{3}\pi R_s^3} = \frac{2}{\sqrt{3}\pi} \approx 0.3676$.
- A hexagonal prism has 8 faces, of which 2 are regular hexagons and 6 are rectangles, 18 edges, and 12 vertices. For a hexagonal prism inscribed within the sensing sphere, all of its 12 vertices are on the spherical surface. The side length of the hexagon is $\frac{\sqrt{6}}{3}R_s$, the height is $\frac{2\sqrt{3}}{3}R_s$, and its volume is $V_h = \frac{3\sqrt{3}}{2}(\frac{\sqrt{6}}{3}R_s)^2\frac{2\sqrt{3}}{3}R_s = 2R_s^3$. The volumetric quotient is $\frac{V_h}{\frac{4}{3}R_s^3} = \frac{3}{2\pi} \approx 0.4775$.
- A rhombic dodecahedron has 12 rhombic faces, 24 edges, and 14 vertices, and all the edges have the same length. The long diagonal of each face is exactly $\sqrt{2}$ times the length of the short diagonal. For a rhombic dodecahedron inscribed within the sensing sphere, only 6 vertices (from the cube A) are on the spherical surface. The edge length is $\frac{\sqrt{3}}{2}R_s$, and its volume is $V_r = 2R_s^2$. The volumetric quotient is $\frac{V_r}{\frac{4}{3}\pi R_s^3} = \frac{3}{2\pi} \approx 0.4775$.
- A truncated octahedron has 14 faces, of which 8 faces are regular hexagonal faces and 6 faces are square faces, 24 vertices, and 36 edges, and all the edges have the same length. For a truncated octahedron inscribed within the sensing sphere, all the vertices are on the spherical surface. The edge length is $\frac{\sqrt{10}}{5}R_s$, and its volume is $V_o = \frac{32\sqrt{5}}{25}R_s^3$. The volumetric quotient is $\frac{V_o}{\frac{4}{3}\pi R_s^3} = \frac{24\sqrt{5}}{25\pi} \approx 0.6833$.

Among the four polyhedrons, the truncated octahedron has the largest volume and volumetric quotient. That is, if we place the nodes to follow the tessellation of truncated octahedrons, then the number of nodes to cover a large space (the volume of the space is much larger than the volume of a sensing sphere) is less than that by following other three tessellations. The critical sensor densities for these four tessellations can be computed as the inverse of the volume of each individual polyhedron.

The node placement can be performed like the growth of a crystal. We take the tessellation of truncated octahedrons for example. First, a node is put at location (x_0, y_0, z_0) (it is better that (x_0, y_0, z_0) is the center of the 3D sensor field). Consider a new (u, v, w) coordinate system with the origin $o = (x_0, y_0, z_0)$. The axes u and v are parallel to the axes x and y, respectively. The unit distance in both the u and v axis is $\frac{4\sqrt{5}}{5}R_s$. The w axis is such that $\angle uow = \angle vow = \arccos(\sqrt{3}/3)$ in the positive quadrant (the axis w creates an angle of $\arccos\sqrt{3}/3$ with the z axis). The unit distance in w axis is $\frac{2\sqrt{15}}{5}R_s$. After creating the u, v, w axes and setting their unit length, the nodes are placed at every integer coordinate of this (u, v, w) coordinate system.

6.1.3 Notes and Comments

We have discussed the node placement for complete 1-coverage. When complete k-coverage (that is, each space point is covered by at least k sensors) is required, a simple approach is to place k sensors at the same location. Another, also simple, approach is to put k layers of tessellations, where each layer of tessellation provides complete 1-coverage. The latter placement method might be more desirable as sensors at different locations are not likely fail (e.g., due to environmental changes) at the same time. As such, it is desirable to place sensors not too close to each other—*minimum separation requirement*—for higher degree of coverage. Kim et al. [15, 16] propose a placement pattern of three layers of regular triangulations to satisfy the minimum separation requirement.

In the three-dimensional placements discussed in the previous subsection, different degrees of network connectivity can be obtained by setting different ranges of communications ranges. On the other hand, for fixed communication range and sensing range, we need to adjust the placement pattern to satisfy both space coverage and network connectivity. Based on the cubic lattice, Bai et al. [8] design the optimal placement patterns to achieve k-connectivity ($k \leq 4$) and 1-coverage for different values of the ratio of communication range over sensing range.

6.2 Random Node Deployment

6.2.1 Vacancy Analysis

Random node deployment is often needed if the sensor field is remote or hostile. Random deployment can be achieved, for example, by scattering sensor nodes from

an aircraft flying over the field of interests. When modeling a random deployment, a sensor node is assumed as a point in a 2D plane (or 3D space), and a random deployment is often modeled as either a uniform point process or a Poisson point process with mean λ (λ is sometimes called the intensity or rate of a Poisson point process). Let $\mathcal{A}$ and $A \doteq \|\mathcal{A}\|$ denote a sensor field and its area, respectively. In a uniform deployment, the nodes' locations are mutually independent random variables, each with uniform distribution in $\mathcal{A}$. That is, each sensor has equal likelihood of falling at any location in $\mathcal{A}$, and independently the other sensors. A uniform N-point process is a uniform deployment of N nodes. Furthermore, the number of nodes within disjoint subregions are mutually independent random variables, and the distribution of the number of nodes within a subregion $\mathcal{A}' \subseteq \mathcal{A}$ is uniform with mean $\frac{\|\mathcal{A}'\|}{\|\mathcal{A}\|} \times N$. A Poisson point process with mean λ is a uniform deployment of λA nodes. Furthermore, the number of nodes within disjoint subregions are mutually independent random variables, and the distribution of the number of nodes within a subregion $\mathcal{A}' \subseteq \mathcal{A}$ is a Poisson with mean $\lambda\|\mathcal{A}'\|$ (see [10], pp. 39–40, for more details).

In the random node deployments of λA (or N) nodes, a sensor field might be completely covered in one random deployment but might not be completely covered in another random deployment. The critical sensor density (CSD), λ_{csd}, helps to decide at least how many nodes are needed to completely cover the sensor field in every random deployment. Given a sensor field with area A, if $\lambda_{\text{csd}} A$ nodes or more are randomly deployed, then the field is completely covered almost surely in each deployment. For the sensing disk coverage model, complete coverage indicates that every point in the sensor field is within the sensing disk of at least one sensor. A field is said to be completely k-covered if every point is within the sensing disks of at least k sensors. The analysis of CSD normally starts by providing bounds for the complete coverage probability of a square field $\mathcal{A}$ with finite area (i.e., $\|\mathcal{A}\|$ is finite) and then uses asymptotical analysis to provide the relation between the CSD and the area (or the side length) of the square. For a field $\mathcal{A}$ with area much larger than the area of a sensing disk (i.e., $A \gg \pi R_s^2$), the boundary effect can be neglected by using a torus convention (cf. [10], p. 23). We can imagine that the sensor field $\mathcal{A}$ is simply one cell of a lattice of squares and all nodes are repeated in precisely the same relative positions in all other cells of the lattice. As illustrated in Fig. 6.5, a sensing disk that protrudes one side of the field $\mathcal{A}$ enters $\mathcal{A}$ again from the opposite side according to such a torus convention.

We use C_λ to denote the event that a square field $\mathcal{A}$ is completely covered by λA sensing disks, each with radius R_s. In the random deployments, the CSD is defined as the smallest possible λ such that for every $\lambda \geq \lambda_{\text{csd}}$, the probability $\Pr[C_\lambda] = 1$ *almost surely*. However, the expression of $\Pr[C_\lambda]$ is not easy to obtain. Instead of seeking for explicit expression of $\Pr[C_\lambda]$, its bound can be obtained by relating the probability of complete coverage to the probability of the field *vacancy* defined as follows. For a point z within the square field $\mathcal{A}$, let $\chi(z)$ denote the indicator function of whether the point z is covered, i.e.,

$$\chi(z) = \begin{cases} 1 & \text{if } z \text{ is not within at least one sensing disk,} \\ 0 & \text{otherwise.} \end{cases} \tag{6.11}$$

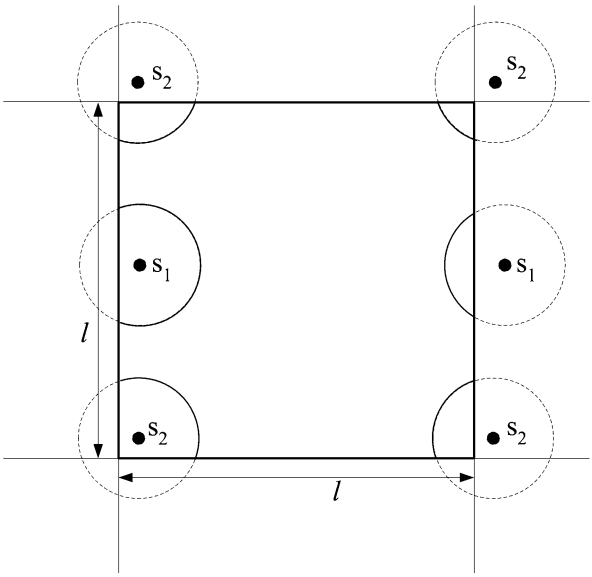

Fig. 6.5 Illustration of the torus convention. A sensing disk that protrudes one side of the field $\mathcal{A}$ enters $\mathcal{A}$ again from the opposite side

The vacancy V within $\mathcal{A}$ is defined as the area that is not covered:

$$V = V(\mathcal{A}) \equiv \int_{\mathcal{A}} \chi(z)\, dz. \tag{6.12}$$

As shown by Hall (see Theorem 3.3 and its remarks in [10]), no vacancy within $\mathcal{A}$ (i.e., $V(\mathcal{A}) = 0$) implies the complete coverage of $\mathcal{A}$. Therefore, it is desirable to provide bounds for probability of no vacancy, i.e., to bound $\Pr[V = 0]$. To do so, the first step is to establish the probability of an arbitrary point to be not covered, i.e., $\Pr[\chi(z) = 1]$, and then to bound the probability of no vacancy based on $\Pr[\chi(z) = 1]$ from some special points in the field. In the literature, two approaches have been used to bound the probability of no vacancy $\Pr[V = 0]$ (or equivalently $\Pr[V > 0]$). One is to exploit the property of the sensing disk model [10, 32, 33]; the other is based on a grid approach [17, 18].

Crossing-Based Approach In the first approach, the special *crossing* points are used to bound the probability of no vacancy $\Pr[V = 0]$. A *crossing* is defined as the intersection point of the outmost circles of two sensing disks (called *sensing perimeter*) or an intersection point of a sensing perimeter and the boundary of the sensor field $\mathcal{A}$. Figure 6.6 illustrates the two kinds of crossings. A crossing is said to be covered if it is an interior point of at least one disk. Note that a crossing is not considered to be covered by its driving sensing perimeters. Hall provides both the lower and upper bounds for $\Pr[V > 0]$ as follows:

Theorem 6.4 (Hall [10], Theorem 3.11) *Suppose that $\mathcal{A}$ is a unit square, and suppose that the random deployment is a Poisson point process with intensity λ. Let*

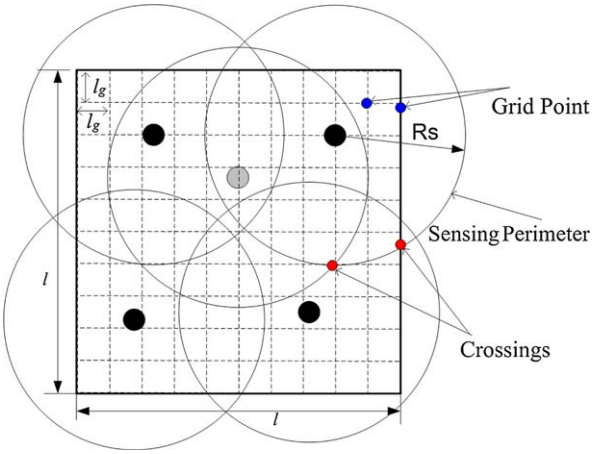

Fig. 6.6 Illustration of using crossing and grid approaches in CSD analysis

$a = \pi R_s^2$. *For all* $\lambda \geq 1$ *and* $0 < R_s \leq \frac{1}{2}$,

$$0.05 \min\left\{1, \left(1 + a\lambda^2\right)e^{-a\lambda}\right\} < \Pr[V > 0] < 3 \min\left\{1, \left(1 + a\lambda^2\right)e^{-a\lambda}\right\}. \quad (6.13)$$

Proof (i) *Upper bound*. In proving the upper bound, Hall claims that the probability of vacancy can be divided into three parts (Theorem 3.11 in [10]):

$$\Pr[V > 0] = p_1 + p_2 + p_3,$$

where

$$\begin{aligned}
p_1 &\equiv \Pr[\text{no disks centered within } \mathcal{A}] \\
&= e^{-\lambda} \\
&= e^{-\lambda(1-a)-a\lambda} \\
&\leq e^{-a\lambda}, \\
p_2 &\equiv \Pr[\text{at least one disk centered within } \mathcal{A}, \\
&\qquad \text{but none of these disks intersects any other disk}] \\
&\leq \Pr[\geq 1 \text{ disk centered within } \mathcal{A}] \\
&\quad \times \Pr[\text{a given disk intersects no other disks}] \\
&= \left(1 - e^{-\lambda}\right)e^{-\lambda\pi(2R_s)^2} \\
&\leq e^{-a\lambda},
\end{aligned}$$

and

$$\begin{aligned}
p_3 \equiv \Pr[&\mathcal{A} \text{ not covered, at least one disk centered within } \mathcal{A}, \\
&\text{and at least one of these disks intersects with another disk}].
\end{aligned}$$

The Markov inequality (see, e.g., [30], p. 263) is applied to upper bound p_3:

$$p_3 \le \Pr[M \ge 2] \le \mathbb{E}[M]/2,$$

where M denotes the total number of uncovered crossings on disks centered within $\mathcal{A}$. The expected number of disks whose centers lie between $R_s + x$ and $R_s + x + dx$ from the center of some given disk $\mathcal{D}$ of radius R_s equals $2\pi(R_s + x)\,dx$; the expected number of crossings of the boundary of $\mathcal{D}$ equals

$$2\int_0^{R_s} \lambda 2\pi(R_s + x)\,dx = 6\lambda\pi R_s^2 = 6a\lambda;$$

the probability that a given crossing is uncovered equals $e^{-a\lambda}$; and the expected number of disks centered within $\mathcal{A}$ equals λ. Therefore,

$$\mathbb{E}[M] = \lambda \cdot 6a\lambda \cdot e^{-a\lambda},$$

whence

$$p_3 \le 3a\lambda^2 e^{-a\lambda}.$$

Therefore, the upper bound follows from the bounds for p_1, p_2, p_3 and the trivial estimate, $\Pr[V > 0] < 3$.

(ii) *Lower bound*. The lower bound is based on the Cauchy–Schwarz inequality.

$$\Pr[V > 0] \ge \frac{(\mathbb{E}[V])^2}{\mathbb{E}[V^2]}.$$

To calculate the expectation, we can use Fubini's theorem [25] and exchange the order of integral and expectation, i.e.,

$$\begin{aligned}
\mathbb{E}[V] &= \int_{\mathcal{A}} \mathbb{E}\big[\chi(z)\big]\,dz \\
&= \int_{\mathcal{A}} \Pr\big[\chi(z) = 1\big]\,dz \\
&= \|\mathcal{A}\| \Pr\big(\chi(z) = 1\big) \\
&= \|\mathcal{A}\| e^{-\lambda\pi R_s^2}, \qquad (6.14)
\end{aligned}$$

where the third equality is due to the fact that $\Pr[\chi(z) = 1]$ is a constant for all z. The computation of $\mathbb{E}[V^2]$ can be divided into two parts:

$$\begin{aligned}
\mathbb{E}\big[V^2\big] &= \mathbb{E}\left[\int_{\mathcal{A}^2} \chi(z_1)\chi(z_2)\,dz_1\,dz_2\right] \\
&= \int\int_{\mathcal{A}^2 \cap \{|z_1 - z_2| \le 2R_s\}} \mathbb{E}\big[\chi(z_1)\chi(z_2)\big]\,dz_1\,dz_2 \\
&\quad + \int\int_{\mathcal{A}^2 \cap \{|z_1 - z_2| > 2R_s\}} \mathbb{E}\big[\chi(z_1)\chi(z_2)\big]\,dz_1\,dz_2. \qquad (6.15)
\end{aligned}$$

If $|z_1 - z_2| = z \le 2R_s$, then

$$\mathbb{E}\big[\chi(z_1)\chi(z_2)\big] = \exp\left[-\lambda\left(2a - R_s^2 B\left(\frac{x}{2R_s}\right)\right)\right],$$

where

$$B(u) \equiv 4\int_u^1 \big(1-y^2\big)^{1/2}\,dy = \pi - 4\int_0^u \big(1-y^2\big)^{1/2}\,dy$$

denotes the area of the lens of intersection of two unit disks centered at $2u$ apart. The second term can be expressed as

$$\begin{aligned}\int_0^u \big(1-y^2\big)^{1/2}\,dy &= (u/2)\big[u^{-1}\arcsin u + \big(1-u^2\big)^{1/2}\big] \\ &\ge (u/2)\arcsin 1 \\ &= (\pi/4)u,\end{aligned}$$

since $u^{-1}\arcsin u + (1-u^2)^{1/2}$ is decreasing in $(0, 1)$. Therefore,

$$\begin{aligned}I_1 &\equiv \int\int_{\mathcal{A}^2\cap\{|z_1-z_2|\le 2R_s\}} \mathbb{E}\big[\chi(z_1)\chi(z_2)\big]\,dz_1\,dz_2 \\ &\le e^{-a\lambda}\int_{\mathcal{A}} dz_1 \int_0^{2R_s} 2\pi z \exp\left[-4\lambda R_s^2\left(\frac{\pi}{4}\right)\left(\frac{z}{2R_s}\right)\right] dz \\ &= 8ae^{-a\lambda}\int_0^1 ye^{-a\lambda y}\,dy.\end{aligned}$$

If $a\lambda \le 1$, we bound the right-hand side by

$$8ae^{-a\lambda}\int_0^1 ye^{-a\lambda y}\,dy = 4ae^{-a\lambda} \le \pi e^{-a\lambda}.$$

If $a\lambda > 1$, we bound it by

$$8ae^{-a\lambda}\int_0^\infty ye^{-a\lambda y}\,dy = 8a^{-1}\lambda^{-2}e^{-a\lambda}.$$

Furthermore, since $\mathbb{E}[\chi(z)] = e^{-a\lambda}$ for all z, and $\chi(z_1)$ and $\chi(z_2)$ are independent for $|z_1 - z_2| > 2R_s$,

$$\begin{aligned}I_2 &\equiv \int\int_{\mathcal{A}^2\cap\{|z_1-z_2|>2R_s\}} \mathbb{E}\big[\chi(z_1)\chi(z_2)\big]\,dz_1\,dz_2 \\ &\le e^{-2a\lambda}.\end{aligned} \tag{6.16}$$

Combining these estimates, we have

$$\frac{(\mathbb{E}[V])^2}{\mathbb{E}[V^2]} = e^{-2a\lambda}/(I_1 + I_2)$$
$$> \begin{cases} e^{-2a\lambda}/(\pi e^{-a\lambda} + e^{-2a\lambda}) & \text{if } a\lambda \le 1, \\ e^{-2a\lambda}/(8a^{-1}\lambda^{-2}e^{-a\lambda} + e^{-2a\lambda}) & \text{if } a\lambda > 1. \end{cases} \tag{6.17}$$

In the case $a\lambda \le 1$, we have

$$\Pr[V > 0] > (\pi e + 1)^{-1} > 0.10 > (0.05)\min\{1, (1 + a\lambda^2)e^{-a\lambda}\}.$$

When $a\lambda > 1$, we have $a\lambda^2 > 1$, and so

$$\begin{aligned} \Pr[V > 0] &> a\lambda^2 e^{-a\lambda}/(8 + a\lambda^2 e^{-a\lambda}) \\ &\ge (1 + a\lambda^2)e^{-a\lambda}/[2(8 + (1 + a\lambda^2)e^{-a\lambda})] \\ &\ge (1/18)\min\{1, (1 + a\lambda^2)e^{-a\lambda}\} \\ &> (0.05)\min\{1, (1 + a\lambda^2)e^{-a\lambda}\}. \end{aligned}$$

This completes the derivation of the lower bound. □

The Hall's method based on crossing points has been extended to k-coverage scenario [27, 33]. Let $\chi_k(z)$ denote the indication function of whether a point z is covered by at most $k-1$ sensors, i.e.,

$$\chi_k(z) = \begin{cases} 1 & \text{if at most } k-1 \text{ nodes cover the point } z, \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, let the k-vacancy V_k denote the area that is covered by at most $k-1$ nodes (i.e., V_k are the area not k-covered). Zhang and Hou [33] argue that the probability $\Pr[V_k > 0]$ can also be decomposed into three parts:

$$\Pr[V_k > 0] = p_1' + p_2' + p_3',$$

where

$$\begin{aligned} p_1' &\equiv \Pr\{\text{no disks centered within } \mathcal{A}\}, \\ p_2' &\equiv \Pr\{\text{at least one disk centered within } \mathcal{A}, \text{ but none of the} \\ &\qquad \text{disks intersects any other disk, and none of the disks} \\ &\qquad \text{intersects the boundary of } \mathcal{A}\}, \\ p_3' &\equiv \Pr\{\mathcal{A} \text{ is not completely } k\text{-covered, at least one disk is} \\ &\qquad \text{centered within } \mathcal{A}, \text{ and at least two disks intersect each} \\ &\qquad \text{other, or at least one disk intersects the boundary of } \mathcal{A}\}. \end{aligned}$$

After providing bounds for $\Pr[V_k > 0]$, asymptotical analysis can be applied to provide the relationship between the sensor density λ and a scaling factor of the sensor field (e.g., the square side length l). Without considering the boundary effect, Zhang and Hou [33] show that the asymptotic k-coverage requires the sensor density λ growing with the side length according to

$$\lambda = \log\left(l^2\right) + (k+1)\log\log\left(l^2\right) + c(l) \tag{6.18}$$

with $\lim_{l\to\infty} c(l) = \infty$. Wang and Yi [27] include the boundary effect into the asymptotic analysis of critical sensor density and derive the asymptotic k-coverage requirement

$$\lambda = \log\left(l^2\right) + 2k\log\log\left(l^2\right) + c(l) \tag{6.19}$$

with $\lim_{l\to\infty} c(l) = \infty$.

Grid-Based Approach In the second approach, a *virtual grid* is created to be embedded within the sensor field, and the grid vertices (or called *grid points*) are used to bound $\Pr[V > 0]$. As shown in Fig. 6.6, a grid with side length l_g is embedded in an $l \times l$ square field. The complete coverage of the square field is linked to the coverage of the grid points of the sensor field by the following claim.

Lemma 6.1 (Kumar et al. [17], Lemma 3.1) *Consider an $l \times l$ square field and a virtual grid $\frac{l}{l_g} \times \frac{l}{l_g}$ embedded in the square field. Let $\mathcal{Z}$ denote the set of all grid points of the grid. If $\mathcal{Z}$ is k-covered by a set of nodes each with sensing radius R_s', then the square field is also completely k-covered by the same set of nodes, each with sensing radius $R_s = R_s' + \frac{\sqrt{2}}{2} l_g$.*

Based on the virtual grid, the probability that all grid points are covered can be lowered bounded by using Janson's Inequality (see, e.g., [4], Theorem 8.1.1) as follows. Let $\mathcal{Z}$ denote the set of grid points. The probability of no vacancy is then defined as the probability that all grid points are covered, which is lower bounded by the product of the probability of all grid point being covered:

$$\Pr[V = 0] \equiv \Pr\left[\bigwedge_{z\in\mathcal{Z}} \big(\chi(z) = 0\big)\right] \geq \prod_{z\in\mathcal{Z}} \Pr\big[\chi(z) = 0\big].$$

Kumar et al. also consider the scenario that each sensor independently becomes active with some probability p, and only active sensors contribute to area coverage. This imposes a Bernoulli-style active process on the Poisson point process, and the resulting process is still a Poisson point process but with a new mean $p\lambda$. Without considering the boundary effect, the probability that a single grid point is uncovered is equal to that there is no sensor falling into the disk $\mathcal{D}$ centered at the grid point, or there are m sensors falling in this disk $\mathcal{D}$ but all are inactive. Kumar et al. further consider the boundary effect and apply asymptotic analysis to derive the relation between sensor density and complete coverage for both uniform and Poisson random deployments.

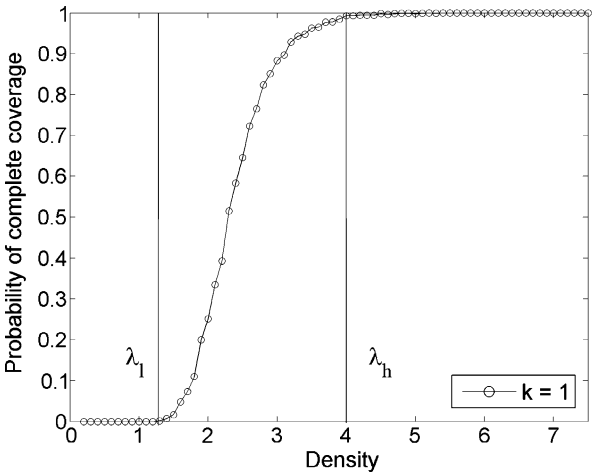

Fig. 6.7 Probability of complete coverage vs. sensor density

6.2.2 Numerical Example

In this subsection, we use computer simulations to obtain the relation between the sensor density and the probability of complete coverage for a sensor field with finite area. We set the sensor sensing radius $R_s = 1$. To reduce any boundary effect, two concentric overlapping square fields with side length of 10 and 12 are used. In each simulation run, we randomly scatter a number of sensors according to a Poisson distribution with mean $\lambda \times 12^2$ within the outer square. The sensor density λ is varied from 0.2 to 7.5 in step of 0.1. A virtual grid with 400×400 vertices is created for the inner square field. These 1.6×10^5 vertices are then examined one by one for their k-coverage. Suppose that in a simulation run, there are m vertices that are not k-covered. Then the vacancy V_k is computed as the $\frac{m}{1.6\times 10^5} \times 100\%$. This process is repeated 1000 times to obtain the average vacancy $\mathbb{E}[V_k]$ for each value of λ. The probability of no vacancy, i.e., $\Pr[V_k = 0]$, is computed as the ratio between the number of runs in which all vertices are k-covered and the number of simulation runs for each simulated density.

Figure 6.7 plots the simulation results of the relation between the sensor density and the probability of complete coverage. A *phase transition* phenomenon can be observed. The transition window is determined by two density values, λ_l and λ_h. λ_l denotes the largest sensor density such that for any sensor density $\lambda \leq \lambda_l$, the probability of complete coverage is 0 almost surely; and λ_h denotes the smallest sensor density such that for any $\lambda \geq \lambda_h$, the probability of complete coverage is 1 almost surely, i.e., the critical sensor density for complete coverage. In other words, if the area of the sensor field is $\|\mathcal{A}\|$ and at least $\lambda_h \|\mathcal{A}\|$ sensors are randomly scattered, then the sensor field can be completely covered almost surely in every random deployment.

Figure 6.8 plots the average k-vacancy ratio against the sensor density, and Fig. 6.9 plots the probability of complete k-coverage against the sensor density. Note that the y-axis of Fig. 6.9 is in log scale. The average 1-vacancy ratio is log

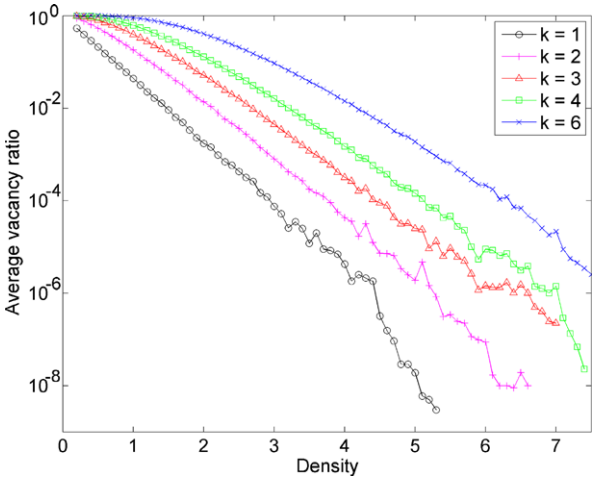

Fig. 6.8 Average k-vacancy ratio vs. sensor density (for the color version, see Color Plates on p. 210)

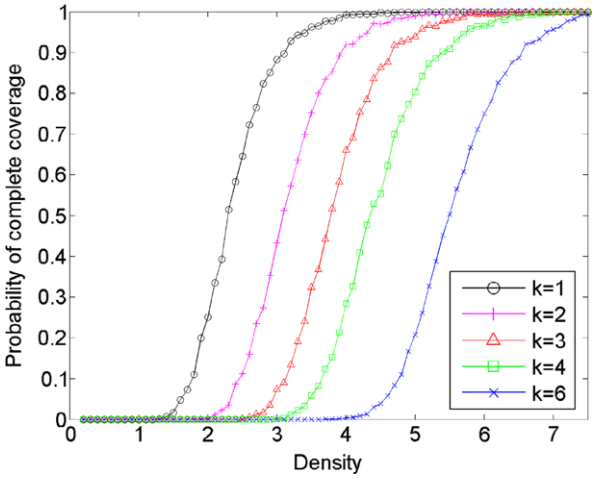

Fig. 6.9 Probability of complete k-coverage vs. sensor density (for the color version, see Color Plates on p. 210)

linear with the sensor density. The small variations in the curve tail are the simulation artifacts due to limited number of simulation runs. When a higher coverage order is used (i.e., k takes larger value), the larger the density is required for complete coverage. Note that only $\mathbb{E}[V] = 0$ implies complete coverage. From the figure we can observe that in some cases when the average k-vacancy ratio is very small, the probability of complete k-coverage is still far from 1. For example, when $k = 1$ and $\lambda = 2$, the average vacancy ratio is 4×10^{-4}, the probability of complete coverage is only 64.6%. Such a small vacancy might not impact on the coverage performance in practice, and sometimes we may use the average vacancy to measure the coverage quality.

6.2.3 Notes and Comments

The analysis of critical sensor density (CSD) provides an insight on the probability of complete coverage in every random deployment. In some cases, one would also like to know the average vacancy if a given number of nodes are to be randomly deployed in a field with finite area [19, 20, 24, 31]. To find the expected value of vacancy, we can use Fubini's theorem [25] and exchange the order of integral and expectation, i.e.,

$$\mathbb{E}[V_k] = \int_{\mathcal{A}} \mathbb{E}\big[\chi_k(z)\big]\,dz = \int_{\mathcal{A}} \Pr\big[\chi_k(z)=1\big]\,dz = \|\mathcal{A}\|\Pr\big[\chi_k(z)=1\big]. \quad (6.20)$$

Compared with the probability of no vacancy $\Pr[V=0]$, the average vacancy $\mathbb{E}[V]$ is easy to obtain. For example, in a random Poisson deployment, an arbitrary point is not covered, if there is no sensor in the disk area centered at this point with radius R_s, i.e.,

$$\Pr\big[\chi(z)=1\big] = e^{-\lambda\pi R_s^2}. \quad (6.21)$$

Similarly, for k-coverage, an arbitrary point is not k-covered if there are less than k sensors in the disk area centered at this point with radius R_s, i.e.,

$$\Pr\big[\chi_k(z)=1\big] = e^{-a\lambda}\left(\sum_{i=0}^{k-1}\frac{(a\lambda)^i}{i!}\right) \quad \text{and} \quad a=\pi R_s^2. \quad (6.22)$$

Yen et al. [31] provide the computation results for the expected coverage ratio in a rectangle sensor field. Their computation also takes the boundary effect into consideration, and the result is summarized in the following theorem.

Theorem 6.5 (Yen et al. [31], Theorem 2) *Suppose that N sensor nodes, each with sensing range R_s, are uniformly distributed at random in an $l \times w$ rectangle ($R_s \le \min(l,w)/2$). Then the expected coverage ratio by these sensors is given by* $1-(1-\frac{\frac{1}{2}R_s^4-\frac{4}{3}lR_s^3-\frac{4}{3}wR_s^3+\pi R_s^2 lw}{l^2w^2})^N$.

We have observed from Fig. 6.7 that a phase transition phenomenon exists in the relation between sensor density and complete coverage. Indeed, the nature of such phase transition is a central topic in the *percolation theory* [9]. Base on the percolation theory, Liu and Towsley [20] argue that the derivation of the average vacancy is independent of $\mathcal{A}$, and hence the average area coverage ratio, $f(\mathcal{A})$, is simply given by

$$f(\mathcal{A}) = 1 - e^{-\lambda\pi R_s^2}. \quad (6.23)$$

Indeed, this result can be extended to arbitrary sensing shapes and for k-coverage. The following theorem by Lazos and Poovendran [19] computes the average area coverage ratio of k-coverage for arbitrary sensing shapes.

Theorem 6.6 (Lazos and Poovendran [19], Corollary 4.10) *Let $\mathcal{S}_i$ denote the sensing shape of a sensor, and let F_i and L_i denote the area and perimeter of the sensing shape, respectively. Consider N homogeneous sensors ($F_i = F$ and $L_i = L$) that are uniformly and independently deployed in a sensor field $\mathcal{A}$ with finite area $A \equiv \|\mathcal{A}\|$ and perimeter L. Suppose that these N sensors cover some part of $\mathcal{A}$. If $\mathcal{A}$ expands in the whole plane in such a way that the sensor density remains a constant ($\frac{N}{A} \to \lambda$), then the fraction of the sensor field covered by at least k sensors, denoted by $f_k(\mathcal{A})$, is given by*

$$f_k(\mathcal{A}) \to \begin{cases} 1, & k = 0, \\ 1 - \sum_{i=0}^{k-1} (\frac{(\lambda F)^i}{i!} e^{-\lambda F}), & k \geq 1. \end{cases}$$

Lazos and Poovendran [19] also consider a more general scenario where each sensor's sensing shape is not fixed but is a random variable drawn from a known probability distribution. For a given sensor field $\mathcal{A}$ of finite area and with N deployed sensors, each with an arbitrary shape of sensing area, what is the fraction of $\mathcal{A}$ being k-covered? This is equivalent to compute the probability that an arbitrary point z in $\mathcal{A}$ being covered by at least k sensors, $\Pr[z \text{ is } k\text{-covered}]$, which is computed by applying the results from *integral geometry* [26]. Since $\Pr[z \text{ is } k\text{-covered}]$ depends only on the volume of each individual sensor's sensing area and the sensing field area as well as their perimeters, their analytical method can be applied to arbitrary node deployment distributions and arbitrary individual sensor sensing shapes.

There are also some CSD analyses based on other sensing models [1, 20, 22, 23, 29]. For example, the critical sensor density for detecting a moving target has been discussed in [1] and [20] and is based on the attenuated sensing model as follows. The sensing intensity exerted by a node s_i to a point z is given by $\frac{S}{d^\alpha(s_i,z)}$, where S is a constant, $d(s_i, z)$ is the Euclidean distance between s_i and z, and α the decay exponent. The sensing intensity experienced by a point z is the sum of the sensing intensity from its nearby nodes or all nodes in the sensor field (e.g., $I(z) \equiv \sum_{i=1}^{N} \frac{S}{d^\alpha(s_i,z)}$). A point z is covered if $I(z)$ is greater than or equal to a threshold, i.e., $I(z) \geq I_{\text{thres}}$. For this sensing model, Adlakha and Srivastava [1] claim that the number of sensors to be deployed in a field $\mathcal{A}$ should be in the order of $O(\frac{\|\mathcal{A}\|}{R^2})$ (R is the upper limit on the sensing region of a single sensor) such that the detection probability for a target moving along a straight line is 98% or above. Liu and Towsley [20] apply the percolation theory and claim the existence of a critical sensor density λ_{csd} such that the detection probability of a target moving across the sensor field is 0 almost surely if $\lambda < \lambda_{\text{csd}}$ and is 1 almost surely if $\lambda \geq \lambda_{\text{csd}}$.

Mo et al. [22, 23] consider a variant of the sensing disk model. Instead of using a fixed sensing radius R_s, they model R_s as a random variable with mean $\bar{R}_s$ and variance $\bar{R}_s^2\sigma_s^2$, and the average sensing area of a sensor is $\bar{a} \equiv \mathbb{E}[\pi R_s^2] = \pi \bar{R}_s^2(1 + \sigma_s^2)$. The derivation of the probability of a point not k-covered and the average k-vacancy are similar to those based on the sensing disk model (i.e., (6.22) and (6.20)), only with the coverage area $a \equiv \pi R_s^2$ of a single sensor being replaced by $\bar{a}$. Wang

et al. [29] define an information coverage for a point based on the cooperative signal estimation. Suppose that an event occurs at a point with constant parameter θ. We can use sensors' measurements to estimate θ. Since the measurements are corrupted by the measurement noise, the estimation $\hat{\theta}$ is modeled as a random variable. A point is said to be (k, ϵ)-covered if its k nearby sensors can estimate the signal parameter occurred on this point within a certain estimation error range, i.e., $\Pr[|\theta - \hat{\theta}| \geq A] \leq \epsilon$. Based on the estimation coverage model, Wang et al. [29] provide an upper bound of a point not (k, ϵ)-covered and use it to bound the average coverage ratio.

References

1. Adlakha, S., Srivastava, M.: Critical density thresholds for coverage in wireless sensor networks. In: IEEE Wireless Communications and Networking Conference (WCNC) vol. 3, pp. 1615–1620 (2003)
2. Akyildiz, I.F., Pompili, D., Melodia, T.: Underwater acoustic sensor networks: Research challenges. Ad Hoc Networks (Elsevier) **3**(2), 257–279 (2005)
3. Alam, S.N., Haas, Z.J.: Coverage and connectivity in three-dimensional networks. In: ACM the 12th Annual International Conference on Mobile Computing and Networking (MobiCom), pp. 346–357 (2006)
4. Alon, N., Spencer, J.: The Probabilistic Method. Wiley, New York (2000)
5. Bai, X., Kumar, S., Xuan, D., Yun, Z., Lai, T.H.: Deploying wireless sensors to achieve both coverage and connectivity. In: ACM the 7th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 131–142 (2006)
6. Bai, X., Xuan, D., Yun, Z., Lai, T.H., Jia, W.: Complete optimal deployment patterns for full-coverage and k-connectivity ($k \leq 6$) wireless sensor networks. In: ACM the 9th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 401–410 (2008)
7. Bai, X., Yun, Z., Xuan, D., Lai, T.H., Jia, W.: Deploying four-connectivity and full-coverage wireless sensor networks. In: IEEE the 27th Conference on Computer Communications (Infocom), pp. 906–914 (2008)
8. Bai, X., Zhang, C., Xuan, D., Teng, J., Jia, W.: Low-connectivity and full-coverage three dimensional wireless sensor networks. In: ACM the 10th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 145–154 (2009)
9. Grimmett, G.: Percolation. Springer, Berlin (1999)
10. Hall, P.: Introduction to the Theory of Coverage Processes. Wiley, New York (1988)
11. Han, X., Cao, X., Lloyd, E.L., Shen, C.C.: Deploying directional sensor networks with guaranteed connectivity and coverage. In: IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON), pp. 153–160 (2008)
12. Kar, K., Banerjee, S.: Node placement for connected coverage in sensor networks. In: International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt) (2003)
13. Sir Thomson, W. (Lord Kelvin): On the division of space with minimum partitional area. Philosophical Magazine **24**(151), 503–514 (1887). URL: http://zapatopi.net/kelvin/papers/on_the_division_of_space.html
14. Kershner, R.: The number of circles covering a set. American Journal of Mathematics **61**(3), 665–671 (1939)
15. Kim, J.E., Yoon, M.K., Han, J., Lee, C.G.: Sensor placement for 3-coverage with minimum separation requirements. In: The 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 266–281 (2008)
16. Kim, J.E., Han, J., Lee, C.G.: Optimal 3-coverage with minimum separation requirements for ubiquitous computing environments. Mobile Networks and Applications (Springer) (2009). doi:10.1007/s11036-008-0122-9

17. Kumar, S., Lai, T.H., Balogh, J.: On k-coverage in a mostly sleeping sensor network. In: ACM International Conference on Mobile Computing and Networking (MobiCom), pp. 114–158 (2004)
18. Kumar, S., Lai, T.H., Balogh, J.: On k-coverage in a mostly sleeping sensor network. Wireless Networks (Springer) **14**(3), 277–294 (2008)
19. Lazos, L., Poovendran, R.: Stochastic coverage in heterogeneous sensor networks. ACM Transactions on Sensor Network (ToSN) **2**(3), 325–358 (2006)
20. Liu, B., Towsley, D.: A study of the coverage of large-scale sensor networks. In: IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), pp. 475–483 (2004)
21. Lyengar, R., Kar, K., Banerjee, S.: Low-coordination topologies for redundancy in sensor networks. In: ACM the 6th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 332–342 (2005)
22. Mo, W., Qiao, D., Wang, Z.: Mostly-sleeping wireless sensor networks: Connectivity, k-coverage, and α-lifetime. In: The 43rd Annual Allerton Conference on Communication, Control, and Computing (2005)
23. Mo, W., Qiao, D., Wang, Z.: Lifetime maximization of sensor networks under connectivity and k-coverage constraints. In: International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 422–442 (2006)
24. Paillard, G., Ravelomanana, V.: Limit theorems for degree of coverage and lifetime in large sensor networks. In: IEEE Infocom, pp. 106–110 (2008)
25. Samko, S.G., Kilbas, A.A., Marichev, O.I.: Fractional Integrals and Derivatives. Gordon and Breach, Yverdon (1993), p. 9
26. Santalo, L.A.: Integral Geometry and Geometric Probability, 2nd Edition. Cambridge University Press, Cambridge (2004)
27. Wan, P.J., Yi, C.W.: Coverage by randomly deployed wireless sensor networks. IEEE Transactions on Information Theory **52**(6), 2658–2669 (2006)
28. Wang, Y.C., Hu, C.C., Tseng, Y.C.: Efficient deployment algorithms for ensuring coverage and connectivity of wireless sensor networks. In: IEEE International Conference on Wireless Internet, pp. 114–121 (2005)
29. Wang, B., Chua, K.C., Srinivasan, V., Wang, W.: Information coverage in randomly deployed wireless sensor networks. IEEE Transactions on Wireless Communications **6**(8), 2994–3004 (2007)
30. Yates, R.D., Goodman, D.J.: Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers. Wiley, New York (1999)
31. Yen, L.H., Yu, C.W., Cheng, Y.M.: Expected k-coverage in wireless sensor networks. Elsevier Ad Hoc Networks **4**(5), 636–650 (2006)
32. Zhang, H., Hou, J.: On deriving the upper bound of α-lifetime for large sensor networks. In: ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 121–132 (2004)
33. Zhang, H., Hou, J.C.: On the upper bound of α-lifetime for large sensor networks. ACM Transactions on Sensor Networks (ToSN) **1**(2), 272–300 (2005)

Chapter 7
Sensor Activity Scheduling

Abstract In one random node deployment, the number of scattered sensors is normally higher than that required by the critical sensor density such that this deployment can achieve complete area coverage almost surely. In some cases, the number of deployed sensor nodes may even be much higher than the optimum to provide certain robustness for the deployed network. After nodes have been deployed, there might be some *redundant sensors* whose covered area can also be covered by other sensors. *Sensor activity scheduling* is used to schedule nodes to be activated alternatively so that the network operation time may be prolonged and certain area coverage requirement can still be met. Generally speaking, the design of a sensor activity scheduling scheme should answer the following question:

How to determine which sensors to be active at which time and be active for how long?

The first part of the question is the focus of this chapter. The second part of the question can be approached by two ways: One is to let each active sensor operate until it depletes its energy, and after that, reselection for active sensors is performed again. The other is to let each active sensor operate for a fixed time interval, and reselection for active sensors is then performed at the beginning of each interval. This chapter first summarizes the assumptions and objectives when designing an activity scheduling scheme and then introduces some representative schemes to illustrate how to select active sensors.

7.1 Assumptions and Objectives

Many activity scheduling schemes proposed in the literature have different assumptions and objectives. The basic assumption is that the coverage model of individual sensors or the covered area of individual sensors is known a priori. Another assumption is on the availability of the nodes' location or distance information: whether each node knows its own Cartesian coordinates, or two neighboring nodes know the Euclidean distance between them. Knowing the nodes locations can be used to derive distances between any pair of nodes, but knowing the distance information

B. Wang, *Coverage Control in Sensor Networks,*
Computer Communications and Networks,
DOI 10.1007/978-1-84996-059-5_7,

may not be enough to derive nodes' locations. If the covered area of individual sensor is not a regular shape (e.g., not a disk), a sensor normally needs its own and its neighbors' locations to decide whether its covered area can also be covered by its neighbors. On the other hand, if the covered area of a sensor is modeled as a disk, a sensor may check its redundancy based only on the distance information. Sometimes, all sensors are assumed to have the same coverage model, and the design of a sensor activity scheduling scheme can be based on the absence of both the nodes' location and distance information.

The basic objective of sensor activity scheduling is to guarantee the *area coverage ratio*, which is defined as the fraction between covered area and uncovered area of a sensor field. We use $\mathcal{A}$ to denote a sensor field and $A(\mathcal{A})$ to denote its area. Let $\mathcal{S} = \{s_1, s_2, \ldots\}$ and $\mathcal{S}_a \subseteq \mathcal{S}$ denote the set of all the deployed sensors and the set of selected active sensors, respectively. Sometimes, $\mathcal{S}_a$ is called a *cover*. Let $A(s)$ denote the area covered by a sensor s. We use $A(\mathcal{S}) = A(\mathcal{A}) \cap (\bigcup_{s \in \mathcal{S}} A(s))$ to denote the area of the sensor field covered by all the deployed sensors. Similarly, we use $A(\mathcal{S}_a) = A(\mathcal{A}) \cap (\bigcup_{s \in \mathcal{S}_a} A(s))$ to denote the area of the sensor field covered by the selected active sensors. A sensor field is completely covered if $A(\mathcal{S}_a) = A(\mathcal{S})$ and is partially covered if $A(\mathcal{S}_a) < A(\mathcal{S})$. The area coverage ratio that is achieved by the selected active sensors is defined as $\frac{A(\mathcal{S}_a)}{A(\mathcal{S})}$.

Generally, we can specify two basic coverage requirements, namely, *complete coverage* and *partial coverage*, for the sensor activity scheduling schemes. Complete coverage requires that the coverage ratio equals one. That is, $A(\mathcal{S}_a) = A(\mathcal{S})$. In most cases, this indicates that a sensor field can also be completely covered by the selected active sensors if all the deployed sensor nodes provide complete area coverage. Partial coverage allows some uncovered area, but it requires that area coverage ratio should be larger than a predefined threshold. That is, $\frac{A(\mathcal{S}_a)}{A(\mathcal{S})} \geq \delta$ where $0 < \delta < 1$.

Another important objective is to select active sensors as least as possible. An active sensor consumes energy to sense physical phenomena and to produce sensing data. Furthermore, its sensing data needs to be sent back to the sink or to be exchanged with other sensor nodes, which increases energy consumption for this active sensor as well as others. Therefore, it is important to reduce the number of active sensors in order to reduce energy consumption and prolong network lifetime. This objective, however, is often in conflict with the coverage ratio objective. In general, the more the active sensors, the higher the coverage ratio. We should make a well balance between coverage ratio and the number of active sensors. This is especially important in the partial coverage requirement.

The network *area coverage lifetime* is also an important objective. Similar to the definition of target coverage lifetime, the area coverage lifetime is defined as the duration from the time that the network starts operation till the time that the area coverage requirement cannot be satisfied even if all the alive sensor nodes are active. Generally speaking, selecting the least number of active sensors helps to prolong the network lifetime. However, care must be taken, since data processing and dissemination also consume energy and impact on the network lifetime. Sometimes, sensor activity scheduling may not lead to any extension of the network lifetime when preserving complete area coverage is required. For example, consider the case that a

small area in a completely covered sensor field is only covered by one sensor. To preserve complete area coverage, this sensor has to be active all the time, and the network lifetime in this example cannot be prolonged via activity scheduling.

There are also some other objectives in the design of a sensor activity scheduling algorithm. For example, the selected active sensors may also be required to form a connected network. The computation complexities, communication overhead, and distributed algorithms are also desirable design objectives. In the rest of this chapter, we discuss the ideas and approaches in the design of activity scheduling schemes, and review some representative scheduling protocols.

7.2 Preserving Complete Area Coverage

In the design of a sensor activity scheduling algorithm to preserve complete area coverage, the first challenge is to determine whether the area covered by one sensor can also be completely covered by its active neighbors. If so, then this node is a redundant one in terms of coverage. A redundant sensor node is eligible to shut off its sensor unit and enters into the energy-saving *sleep* state. However, the redundancy of a node depends on its neighbors' states. A node may be no longer redundant if one of its active neighbors becomes inactive. The second challenge is to determine the order of sensor nodes' activation or deactivation. In this section, we introduce several redundancy check methods, discuss two activity scheduling procedures, and elaborate two representative scheduling protocols.

7.2.1 Redundancy Check Methods

A straightforward method for redundancy check is to use a grid approach, where each sensor maintains a list of grid points within its covered area, as shown in Fig. 7.1. If such grid points are covered by its active neighbors, then it is a redundant one. For example, in Fig. 7.1, the grid points marked by small blue circles are covered by the sensor s_1, and all these grid points are also covered by its neighbors. Hence, s_1 is redundant if its neighbors s_2, s_3, s_4, and s_5 are active. Using the grid approach is a general method, as it does not restrict the sensing coverage model used by individual sensors. Instead, it needs to specify the coverage relation between each grid point and each sensor, which, however, may be computation complicated, time consuming, and storage expensive. For the sensing disk coverage model, some of its geometric properties can be exploited for redundancy check, and computation complexity and storage requirement can be greatly reduced. Many redundancy check methods have been proposed in the literature, and we briefly introduce some representative ones in what follows.

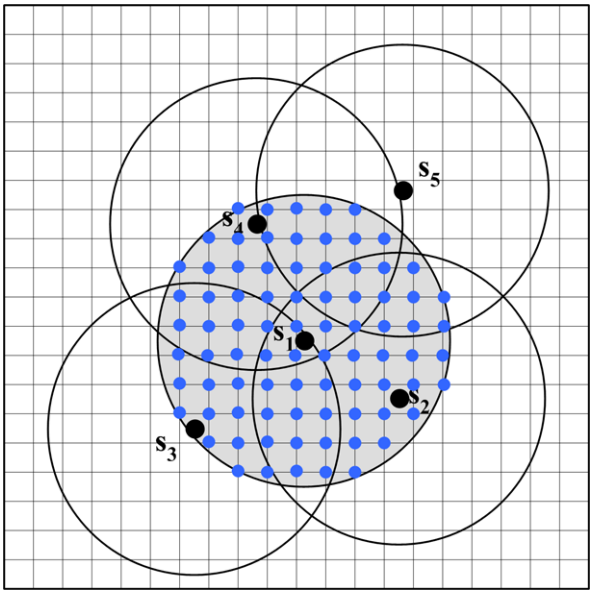

Fig. 7.1 Illustration of using grid approach for redundancy check

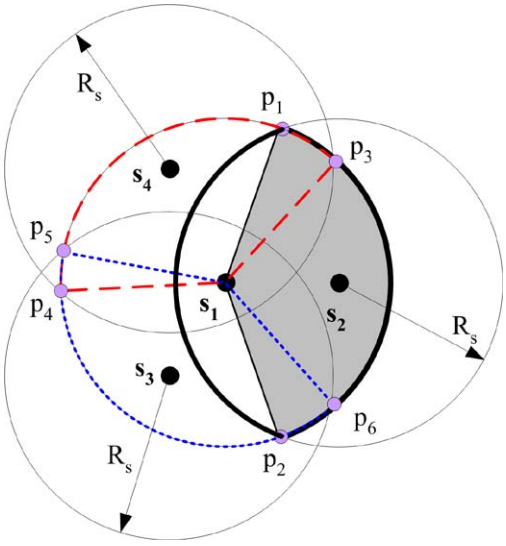

Fig. 7.2 Sponsored sector coverage for redundancy check

Sponsored Sector Tian and Georganas [38] propose a concept of *sponsored sector* for checking redundancy. A sensor s_2 is called a sponsor to sensor s_1 if $d(s_1, s_2) \leq R_s$, where $d(s_1, s_2)$ denotes the Euclidean distance between s_1 and s_2. Since $d(s_1, s_2) \leq R_s$, the two sensing disks intersect. As shown in Fig. 7.2, the crescent-shaped area bounded by the bold arcs is the intersection of the two sensing disks. The sponsored sector by sensor s_2 to sensor s_1 is the sector area of s_1 within the intersection crescent of the two sensing disks, as shown by the shaded sector in

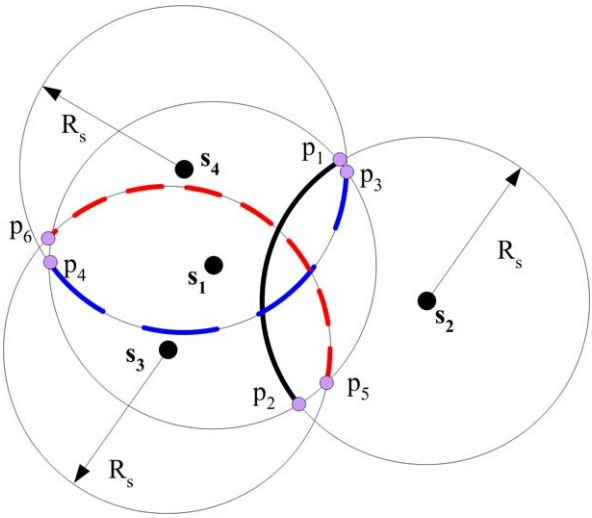

Fig. 7.3 Perimeter coverage for redundancy check

Fig. 7.2. The reason to use the sponsored sectors other than the intersection crescents for redundancy check is to reduce calculation complexity. The area of a sector can be represented by its central angle (e.g., $\angle p_1 s_1 p_2$ in the figure) accurately, and uniting the areas of two sectors is equivalent to merging two central angles. The redundancy rule by sponsored sector is as follows: *If the sensing disk of a sensor s can also be covered by the sponsored sectors from its sponsors, then the sensor s is redundant and eligible to be inactive.* In Fig. 7.2, the sensing disk of sensor s_1 is covered by the three sponsored sectors contributed from the sensors s_2, s_3, and s_4, and hence sensor s_1 is redundant and eligible to go to the sleep state. Since the area of the sponsored sector is smaller than that of the crescent, it is expected that the overlapped area by the selected active sensors is large, and hence more active sensors will be selected.

Perimeter Coverage Huang et al. [27] propose to use *perimeter coverage* to check sensor redundancy. Two sensors' sensing disks intersect each other if $d(s_1, s_2) < 2R_s$, and each sensor is called a direct neighbor to the other. As illustrated in Fig. 7.3, sensor s_1 has three direct neighbors. The arc $\overset{\frown}{p_1 p_2}$ is a segment of s_2's sensing perimeter within s_1's sensing disk and is covered by sensors s_3 and s_4. In this regard, s_1 is called a candidate for its director neighbor s_2. The redundancy rule by perimeter coverage is as follows: *If a sensor s is a candidate for each of its direct neighbors, then it is redundant and eligible to be inactive.* In Fig. 7.3, sensor s_1 is a candidate for its direct neighbor s_3 (the arc $\overset{\frown}{p_5 p_6}$ covered by s_2 and s_4) and also a candidate for its direct neighbor s_4, and hence it is redundant and eligible to go to the sleep state. The perimeter coverage method can also be applied for k-coverage redundancy check. The sensing disk of a sensor is k-covered if every point within its sensing disk is covered by k distinct sensors. For k-coverage redundancy check, a sensor s_1 is a candidate for its direct neighbor s_2 if the arc $\overset{\frown}{p_1 p_2}$ is covered

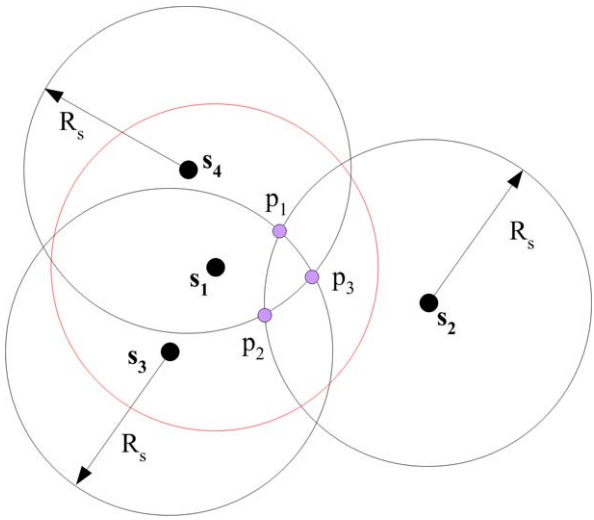

Fig. 7.4 Crossing coverage for redundancy check

by k distinct sensors other than the sensor s_1. The same perimeter coverage rule then applies for determining whether a sensor is k-covered by its direct neighbors.

Crossing Coverage Xing et al. [51] apply *crossing coverage* to determine redundant sensors. If two sensing disks intersect each other, then they create crossings that are the intersection points on the two disks' perimeters. As shown in Fig. 7.4, the crossing p_1 within s_1's sensing disk is created by s_2's sensing perimeter and s_3's sensing perimeter. Crossing points can also be caused due to the intersection between a sensor's sensing perimeter and the sensor field boundary. The points on the sensing perimeter of a sensor is not considered as covered by this sensor. Therefore, in Fig. 7.4, the crossing p_1 is not covered by the sensors s_2 and s_3. But p_1 is covered by the sensor s_4. In this regard, the crossing p_1 is considered as a covered crossing within the sensor's s_1 sensing disk. The redundancy rule by crossing coverage is as follows: *If all crossings within the sensor's s sensing disk are covered, then s is redundant and eligible to be inactive.* In Fig. 7.4, there are three crossings (p_1, p_2, and p_3) within the sensor's s_1 sensing disk, and they are all covered (p_2 covered by s_3, and p_3 covered by s_2), and hence s_s is redundant and eligible to be inactive. The crossing coverage method is easily extended to k-coverage redundancy check. A crossing is k-covered if it is covered by k distinct sensors. A sensor is k-covered and redundant if all crossings within its sensing disk are at least covered by k distinct sensors excluding itself.

Voronoi Diagram Vertices and Intersections Cărbunar et al. [8] propose to use *Voronoi diagram vertices and intersections* to check redundancy. A Voronoi diagram for N sensors $s_1, s_2, \ldots, s_N$ in a plane is defined as the subdivision of the plane into N cells, each for one sensor, such that the distance between any point in a cell and the sensor of the cell is closer than that between this point and other sensors. Two Voronoi cells meet along a *Voronoi edge*, and a sensor is a *Voronoi*

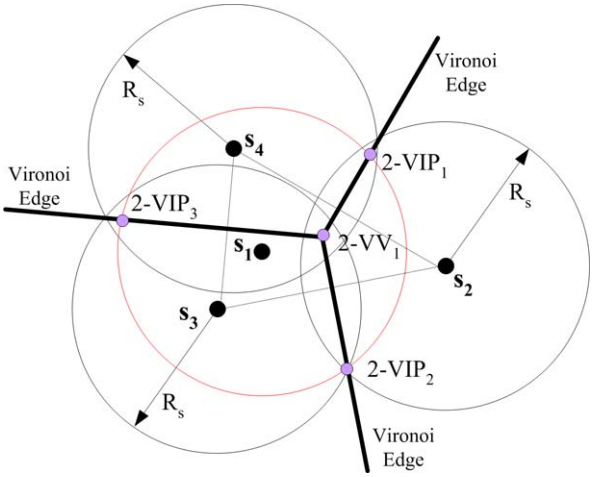

Fig. 7.5 Voronoi vertices and intersections coverage for redundancy check

neighbor of another sensor if they share a Voronoi edge. When checking sensor s redundancy, a 2-Voronoi diagram is first constructed, which is the Voronoi diagram of the Voronoi neighbors of s when s is excluded. In Fig. 7.5, the bold lines form a 2-Voronoi diagram for sensor s_1. The 2-Voronoi Vertices (2-VV) of a sensor s are the Voronoi vertices of the 2-Voronoi diagram of s. A 2-Voronoi Intersection Point (2-VIP) of s is the intersection between an edge of the 2-Voronoi diagram and the sensing perimeter of s. In Fig. 7.5, there are one 2-VV (2-VV_1) and three 2-VIPs (2-VIP_1, 2-VIP_2, 2-VIP_3). The redundancy rule by Voronoi diagram vertices and intersections is as follows: *If all the 2-VVs and 2-VIPs of a sensor s are covered by the Voronoi neighbors of s, then s is redundant and eligible to be inactive.* In Fig. 7.5, sensor s_1 is eligible since 2-VV_1 and $2\text{-VIP}_{1,2,3}$ are all covered by s_1's Voronoi neighbors.

7.2.2 Activity Scheduling Procedures

A sensor activity scheduling algorithm can be either centralized or distributed. A distributed scheduling algorithm is more desirable, as it can be easily scaled to large-scale sensor networks. Most existing protocols are distributed, and only local message exchanges are incurred in the sensors' decision process. Although these protocols differ in their redundancy check methods, their scheduling procedures are similar. It is often assumed in these protocols that the time-line is divided into consecutive rounds. At the beginning of each round, there is a decision stage where all sensor nodes should make their activity decisions in the current round. Normally, the length of the decision stage is much less than the length of a round. At the end of a round, all sensors are required to be active, and they enter the decision stage again in the next round. Sensor nodes are normally required to be time-synchronized so that they can be back to active at the end of each round at almost the same time.

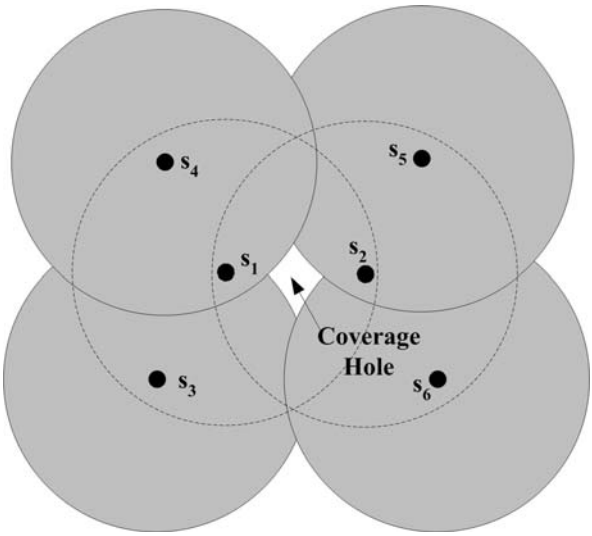

Fig. 7.6 Illustration of a coverage hole caused by the inactivation of two redundancy-dependent sensors

Generally speaking, there are two approaches for sensor nodes making activity decision in each decision stage, *self-inactivation* approach and *sequential activation* approach. The two approaches differ in the message exchange and process method. In the self inactivation approach, a sensor which has decided to be inactive broadcasts a SLEEP message that is used to alert its neighbors to recheck their redundant eligibility. In the sequential activation approach, a sensor which has decided to be active broadcasts an ACTIVE message that is used to set or reset its neighbors' activation timers. Many protocols have included some more message types and have introduced transient states and transition machines to avoid simultaneous inactivations and to combat transmission collisions and losses. In what follows, we simply use two states, i.e., *active* and *sleep*, and two message types, i.e., SLEEP and ACTIVE, to describe the basic operation steps of the two approaches.

Self-Inactivation In the self-inactivation approach, each sensor node maintains a list of its active neighbors. At the beginning of each decision stage, the list actually includes all of its neighbors. A sensor node performs redundancy check based on the assumption that all neighbors in the list are active. After a sensor node has decided its own redundancy, it can go into a sleep state. When this is done in a distributed manner, care must be taken to avoid creating *coverage holes*. Since a sensor redundant eligibility depends on its neighbors' active state, a coverage hole may appear if two redundancy-dependent sensor nodes become sleep at the same time. For example, in Fig. 7.6, s_1 is completely covered by s_2, s_3, and s_4, and s_2 is also completely covered by s_1, s_5, and s_6. However, s_1 and s_2 are redundancy-dependent sensors and cannot be simultaneously inactive. Otherwise, some points only covered by s_1 or s_2 will not be covered, and a coverage hole is created. This problem can be mitigated or avoided by using a random backoff mechanism and carefully designed message exchange process for asynchronous activity decision-making. We sketch

the basic steps required in the decision stage of the self-inactivation approach as follows.

Steps of the Self-Inactivation Approach

Step 0: Set self state as *active*.
Step 1: Collect all neighbors' information via local message exchanges and build a list of active neighbors.
Step 2: Perform redundancy check. If redundant, set a timer with a random backoff time and goto Step 3.
Step 3: Wait for the timer expiration. If receive a SLEEP message, rebuild the list of active neighbors and goto Step 2. If timer expires, goto Step 4.
Step 4: Broadcast a SLEEP message and set self state as *sleep*.

Sequential Activation In the sequential activation approach, each sensor node also maintains a list of its active neighbors. Compared with the self-inactivation approach, the list is empty at the beginning of each decision stage. Each sensor sets an activation timer. Upon the expiration of the timer, it sets itself as active and broadcasts an ACTIVE message. A sensor node which has received a new ACTIVE message resets its timer, builds its active neighbor list, and performs redundancy check. The redundancy check is normally based on the grid approach. A redundant sensor sets itself as sleep and ignores the following messages. As the name suggests, it is desirable that sensor nodes are activated sequentially. First, a sensor volunteers to be active with a small probability, and a nonvolunteer sets a long expiration time. This first active sensor node then activates its neighbors by broadcasting an ACTIVE message to reset its neighbors' timers, and such process continues until all sensors have decided their states. How to adjust the activation timer determines the selection of desired active sensors and the number of active sensors. We sketch the basic steps of the sequential activation approach as follows.

Steps of the Sequential Activation Approach

Step 0: Set self state as *active*, and set an empty list of active neighbors.
Step 1: Volunteer to be active with a small initial probability. If volunteer, goto Step 4. If not, set a timer and adjust the volunteer probability, and goto Step 3.
Step 2: Perform redundancy check. If redundant, set self state as *sleep*. If not, goto Step 3.
Step 3: Wait for the timer expiration. If receive an ACTIVE message, rebuild the list of active neighbors, adjust the expiration timer, and goto Step 2.
Step 4: Broadcast an ACTIVE message.

7.2.3 Example Scheduling Protocols

We elaborate two representative sensor activity scheduling protocols: One is the coverage configuration protocol (CCP) [51], which applies crossing-based redundancy check and follows the self-inactivation scheduling procedure. Another is the

optimal geographical density control (OGDC) [57], which applies grid points-based redundancy check and follows the sequential activation scheduling procedure.

Coverage Configuration Protocol (CCP) In CCP, all nodes are initially in the ACTIVE state. A node decides to go to the SLEEP state if all of the crossings within its sensing disk can also be covered by its active neighbors. As such a decision making is done in a distributed way, some other transient states and backoff timers are used to avoid creating coverage holes due to simultaneous inactivation of neighboring nodes. In CCP, five states are used, namely, the ACTIVE, SLEEP, LISTEN, JOIN, and WITHDRAW states. The transition rules are outlined as follows.

- In SLEEP. When the sleep timer T_s expires, a node turns on the radio, starts a listen timer T_l, and enters the LISTEN state.
- In LISTEN. When a message (HELLO, WITHDRAW, or JOIN message) is received, a node evaluates its eligibility of becoming active. If it is eligible, it starts a join timer T_j and enters the JOIN state. Otherwise, it sets a sleep timer T_s and returns to the SLEEP state when T_l expires.
- In JOIN. If a node becomes ineligible of becoming active before T_j expires (e.g., due to the reception of a JOIN message), it cancels T_j, starts a sleep timer T_s, and returns to the SLEEP state. If T_j expires, it broadcasts a JOIN message and enters the ACTIVE state.
- In ACTIVE. When a node receives a HELLO message, it executes the crossing-based redundancy check to determine its eligibility to remain active. If it is ineligible to be active, it starts a withdraw timer T_w and enters the WITHDRAW state.
- In WITHDRAW. If a node becomes eligible (due to the reception of a WITHDRAW or HELLO message from a neighbor) before the T_w expires, it cancels the T_w and returns to the ACTIVE state. If T_w expires, it broadcasts a WITHDRAW message, starts a sleep timer T_s, and enters the SLEEP mode.

Both the join and withdraw timers are randomized to avoid collisions among multiple nodes that decide to join or withdraw at the same time. The values of these timers impact on the protocol performance. They can be set according to the network density or the node's coverage efficiency (how much uncovered area can be covered if this node is active).

Optimal Geographical Density Control (OGDC) The basic idea behind the OGDC is to dynamically emulate a triangular tessellation process. It is well known that putting disks centered at the vertices of an equilateral triangle lattice requires the least number of disks to provide complete coverage for a 2D plane [29]. In OGDC, each newly activated node specifies the next desired location best approximating the vertex in a triangular tessellation in order to activate the next sensor, and the process continues until all nodes have decided their states.

In OGDC, three states are used, namely, the ACTIVE, SLEEP, and UNDECIDED states. The UNDECIDED state is a transient state. The procedure of OGDC can be divided into two phases: One is the volunteer phase, where a node decides

whether or not to be a starting active node; the other is the decision phase, where a node decides, upon reception of an ACTIVE message, whether or not to become an active node. In what follows, we provide a brief description of the OGDC protocol.

Volunteer Phase At the beginning of each round, all nodes are in the UNDECIDED state. A node volunteers to be a starting node with probability p if its residual energy exceeds a predetermined threshold P_t. If a sensor node volunteers, it sets a volunteer timer T_d with the value drawn from a uniform distribution. When the volunteer timer expires, this node changes its state to the ACTIVE state, and it broadcasts an ACTIVE message. If a node hears other ACTIVE messages before its volunteer timer expires, it cancels its timer and does not become a starting node. The ACTIVE message by the starting node contains (i) the position of the sender and (ii) the direction α along which the second desired node should be located. This direction is randomly generated from a uniform distribution in $[0, 2\pi]$. Nonstarting node may also send ACTIVE message, and in this case, the direction field is set to -1 to indicate that the sender is not a starting node.

If a node does not volunteer itself to be a starting node, it sets a starting timer T_s. Upon the expiration of T_s, it repeats the above volunteering process with p doubled until its value reaches one. The starting timer T_s is canceled, whenever the state of a node is changed to ACTIVE or SLEEP in response to other ACTIVE messages.

Decision Phase When a sensor node receives an ACTIVE message, if the node is already in the active state, or it is more than $2R_s$ away from the sending node, it ignores the message; otherwise, it adds this sender to its active neighbor list and checks whether or not all its active neighbors can completely cover its own coverage disk. If so, the node sets its state to the sleep state.

If the node does not ignore the ACTIVE message and its active neighbors do not completely cover its coverage disk, it enters one of the following three cases, as depicted in Fig. 7.7: (i) There exists an uncovered crossing that is created by its active neighbors and falls in the node's coverage disk; (ii) the condition in (i) is not satisfied, and at least one neighbor is a starting node; (iii) neither (i) nor (ii) satisfies. A node can determine if a neighbor is a starting node from the direction field of the ACTIVE message sent by that neighbor (a positive value indicates a starting node, and –1 indicates a nonstarting node).

In case (i), the node first finds the closest uncovered crossing that falls in its coverage disk. If the closest uncovered crossing is created by the new neighbor that sends the latest ACTIVE message to the node, the node will cancel the existing timer (T_{c1}, T_{c2}, or T_{c3}, if any) and reset the timer T_{c1}. Otherwise, the node retains the existing timer. The rationale behind how the value of T_{c1} is calculated is illustrated in Fig. 7.8. After node A has activated node B, the next optimal sensor to be activated is the one located at the optimal position O (if there is) such that $\triangle ABO$ can form an equilateral triangle. If there is no such a sensor at the position O, a sensor with least deviation from such an optimal location will be likely to be first activated. In Fig. 7.8, a node R receives the active messages from A and B. Let d denote the distance between the receiver node and the crossing point, and $\Delta\alpha$ the angle between

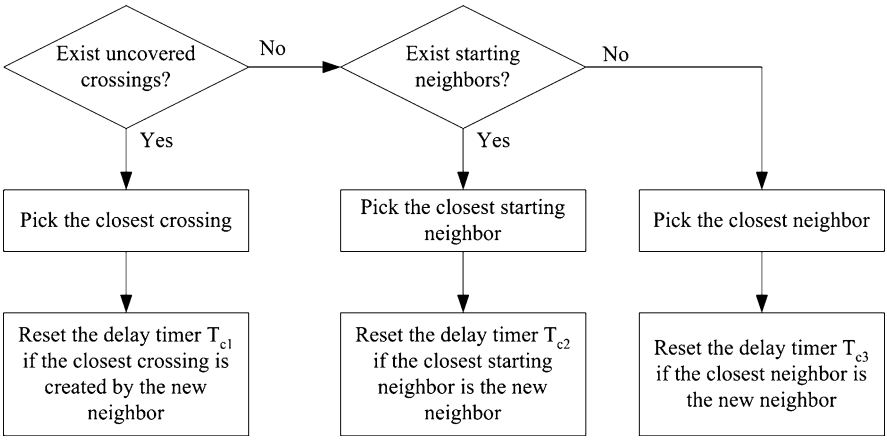

Fig. 7.7 The procedure taken when a node receives an ACTIVE message in OGDC (reproduced from [57], ©2005, Old City Publishing)

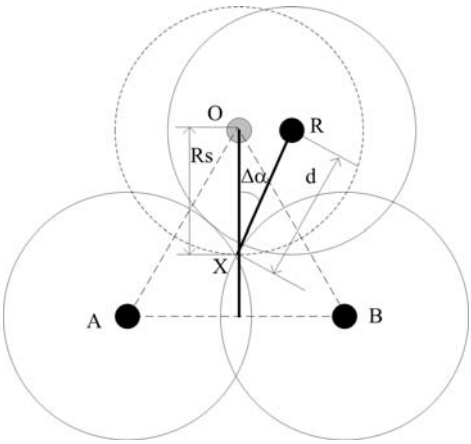

Fig. 7.8 Illustration of OGDC triangular tessellation process and timer setting (in case (i)) (reproduced from [57], ©2005, Old City Publishing)

$\overline{XO}$ and $\overline{XR}$. The value of T_{c1} is set as

$$T_{c1} = t_0\big(c\big((R_s - d)^2 + (d\Delta\alpha)^2\big) + u\big),$$

where t_0 is the transmission time of an ACTIVE message, c is a constant determining the backoff scale, and u is a random number drawn from the uniform distribution in [0, 1]. T_{c1} includes two terms: a deterministic term $c((R_s - d)^2 + (d\Delta\alpha)^2)$ and a random term u. If the receiver is right in the direction α and its distance to the crossing is R_s, the deterministic term is 0; otherwise, it roughly represents the deviation from the optimal position, and a delay is introduced in proportion of this deviation. The random term is introduced to break ties in the case that there exist nodes whose locations yield the same value of the deterministic term.

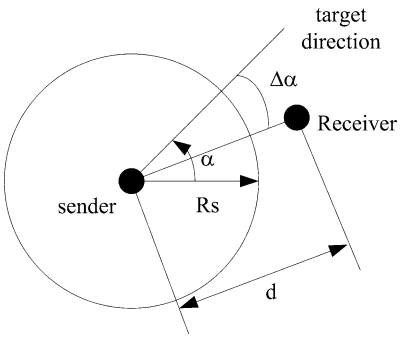

Fig. 7.9 Illustration of OGDC timer setting (in case (ii)) (reproduced from [57], ©2005, Old City Publishing)

In case (ii), the node finds the closest starting neighbor. If the closest starting neighbor is the new neighbor, the node cancels the existing backoff timer (T_{c1}, T_{c2}, or T_{c3}, if any) and resets a backoff timer T_{c2}. Otherwise, the node retains the existing timer. As illustrated by Fig. 7.9, the value T_{c2} is set as

$$T_{c2} = t_0\big(c\big((\sqrt{3}R_s - d)^2 + (d\Delta\alpha)^2\big) + u\big),$$

where t_0, c, u are the same as above, d is the distance from the sender to the receiver, $\Delta\alpha$ is the angle between α and the direction from the sender to the receiver.

In case (iii), the node finds the closest neighbor. If the closest neighbor is a new neighbor, the node cancels the existing backoff timer (T_{c1}, T_{c2}, or T_{c3}, if any) and resets a backoff timer T_{c3}, which is much greater than that of the average values of T_{c1}, T_{c2} but much less than the value of T_s. Otherwise, it retains the existing timer.

In any of the above three cases, when the backoff timer expires, the node sets itself to active state and broadcasts an ACTIVE message with the direction field α set to –1 (indicating the message sent by a nonstarting node).

7.2.4 Notes and Comments

The grid approach is the basic method for redundancy check and has been widely used in many protocols. Moreover, the grid approach can also be used in sensor activity scheduling to achieve *differentiated coverage*, where each grid point may be required to be covered by different number of sensors [12, 52]. There are also some extensions or variants of the four redundancy check approaches discussed in this section, such as the extended sponsored area approach [7, 28, 34], the extended perimeter coverage approach [35, 42], the extended crossing approach [15, 26, 33], and the extended Voronoi approach [6]. These schemes have extended the basic redundancy check approaches in order to schedule fewer active sensors, to reduce computation complexities or to cope with sensors with different sensing ranges.

The redundancy check approaches discussed in this section, including the grid approach and the four sensing disk properties-based approaches, require the location information of each sensor node. Some researchers argue that network-wide

node localization may be unnecessary (or possibly infeasible) for coverage control and propose to use estimated distances between neighboring nodes, instead of their coordinates, to schedule sensor activity [48, 55, 58]. Nodes distance information can be obtained from the received signal strength based on an attenuation radio transmission model or estimated from the number of neighboring nodes, given that the deployment density is known to every node. For example, the basic idea behind [48] is to guarantee that all points within a triangle are covered by the three active sensors on the triangle vertices.

There is another interesting application of sensing disk perimeter coverage. For a given sensor network, how to efficiently decide whether or not that every point of the sensor field can be covered by at least k sensors? Huang and Tseng [24] present a polynomial-time solution, in terms of the number of sensor nodes, to this decision problem. Their solution is based on the sensing disk perimeter coverage. A sensor is called k-perimeter-covered if every point on its sensing disk perimeter is within the sensing area of at least k other sensors. The sensor field is completely k-covered iff each sensor in the network is k-perimeter-covered. Huang et al. [25] also extend their result to three-dimensional networks. Note that this perimeter coverage check is different from the perimeter coverage check for node coverage redundancy. In the node redundancy check, the arcs of other sensor's sensing disk perimeter within a sensor's sensing disk are examined.

7.3 Preserving Partial Area Coverage

Preserving complete area coverage is a desirable yet demanding objective in sensor activity scheduling. Sometimes, an activity scheduling protocol that can provide high average coverage ratio may be of more practical interests. *Network coverage lifetime* is an operational measure for a sensor network and is often defined as the duration from the network setup time to the time that the network coverage cannot be guaranteed when all sensors are active. Analysis and simulations (e.g., [44, 56]) have shown that the network coverage lifetime can be greatly prolonged if only preserving partial coverage other than preserving complete coverage is required.

Normally, a sensor node does not need to perform a redundancy check for its own complete coverage when only preserving partial coverage is required. Hence computation complexity and storage requirement can be greatly reduced. Many activity scheduling protocols have been proposed for preserving partial area coverage. We classify them into two main groups. In the first group, a sensor makes its activity decision independent of others. In the second group, a sensor exploits its neighbors' information to make activity decision.

7.3.1 Random Independent Sleeping

Random Independent Sleeping (RIS) might be the simplest sensor activity scheduling scheme where a sensor node decides its activity states independently of other

sensor nodes (see, e.g., [1, 9, 17, 39]). RIS has two main advantages: (1) No location or distance information is required; and (2) no control message is required in RIS. A RIS scheme can be implemented in either asynchronous or synchronous approaches.

An asynchronous approach can be as follows. The time line is divided into consecutive rounds with equal length T for each sensor node, but the beginning time of the very first round is different across nodes, i.e., the rounds are not synchronized across sensors. At the beginning of a round, a sensor decides its active state with the duration given by $p \times T$, and the remaining part of the round is the sleep state.

Another approach to implement a RIS scheme is to use synchronous decisions as follows. The time line is divided into rounds of equal length, and the starting time of every round is considered to be synchronized across sensor nodes. Two versions can be implemented following this synchronized decision approach. One is to let, at the beginning of each round, each node to decide its active state for this round with probability p. This method and the aforementioned asynchronous RIS produce nondisjoint sets of active sensors in different rounds. That is, the intersection between the active sensor set produced in round i and that produced in round $i+1$ may not be an empty set. Another version is to divide the deployed nodes into K disjoint subsets to be activated in a round-robin manner. Each of such subsets is called a *cover*, and how to find a partition to produce $K \geq 2$ disjoint covers is called Set K-Cover problem [1]. A simple randomized algorithm is to let each sensor randomly generate an integer between 1 and K to decide which set it belongs to. Obviously, this method produces disjoint subsets of active sensors in different rounds.

Suppose that N sensor nodes are uniformly deployed in the sensor field. The expected number of active sensors at any time in RIS is $p \times N$ or N/K, and they are distributed uniformly. The parameter p or K in the RIS is assumed as globally known by all sensors, and its value depends on the requirements of coverage ratio. Although RIS is very easy to be implemented, it may lead to low coverage ratio (when p is small or K is large) or more selected active sensors (when p is large or K is small). Analysis or simulation can be used to determine the relation between the values of p or K and the coverage ratio by the randomly activated sensors.

The relation between the number of deployed sensors N and the area coverage ratio ρ_c can be derived by using asymptotic analysis as follows [20, 31]. Let λ denote the sensor density, i.e., the number of sensor nodes per unit area. For a given large region $\mathcal{A}$ with area A, the number of nodes in this area is $n = \lambda A$. The probability of a point within $\mathcal{A}$ which is not covered by an arbitrary sensor (i.e., the point does not lie within the sensor's sensing disk) is given by $1 - \frac{a}{A}$, where $a = \pi R_s^2$. Since sensors are uniformly located in $\mathcal{A}$, the probability that the point is not within any sensor equals $(1 - \frac{a}{A})^n$. In the limit as $A \to \infty$, we have

$$\begin{aligned} \Pr(\text{a point not covered}) &= \mathbb{E}\left[\left(1 - \frac{a}{A}\right)^n\right] \\ &= e^{-(a/A)\mathbb{E}(n)} \\ &= e^{-a\lambda} = e^{-\lambda\pi R_s^2}. \end{aligned} \tag{7.1}$$

The fraction of the area being covered is hence given by

$$\rho_c = 1 - e^{-\lambda\pi R_s^2}. \tag{7.2}$$

This formula can be used in the network planning to determine the required sensor density. Given the sensor density and the area of the sensor field, it can be used to determine the active probability p or the number of covers K for a specified coverage ratio.

In RIS, a point that is within K sensors' sensing disks may be k-covered ($k \leq K$) by at least k active sensors in some time interval (t_1, t_2) and not k-covered at all in another time interval (t_2, t_3). A point is not k-covered if it is covered by less than k active sensors. The coverage of a point hence can be modeled as an *alternative renewal process* with two states, k-covered and not k-covered. The probability or the distribution of such a point being k-covered is important to detection and tracking applications. Let $Y^{(k)}$ and $Z^{(k)}$ be two random variables denoting the length of k-covered period and not-k-covered period, respectively. Given that a point is within n sensors' sensing disk and that the sensors are active with probability p in each round, the expectations for $Y^{(k)}$ and $Z^{(k)}$ can be computed as follows [23]:

$$\mathbb{E}\left[Y^{(k)}\right] = \frac{1 - \sum_{i=0}^{k-1} \binom{K}{i} p^i (1-p)^{K-i}}{K \binom{K}{k-1} p^k (1-p)^{K-k+1}}, \quad k = 1, \ldots, K,$$

and

$$\mathbb{E}\left[Z^{(k)}\right] = \frac{\sum_{i=0}^{k-1} \binom{K}{i} p^i (1-p)^{K-i}}{K \binom{K}{k-1} p^k (1-p)^{K-k+1}}, \quad k = 1, \ldots, K.$$

Hsian and Liu [22] also analyze the tail distribution of the probability that a point within n sensors' sensing disks cannot be covered by any active sensor.

7.3.2 Neighbor Based Scheduling

When only statistical coverage ratio rather than complete coverage is required, a sensor node normally needs not to perform redundancy check for decision making of active state. Unlike the totally independent RIS scheduling, a sensor node can schedule its activity based on its neighbors' information. Such information includes the distances between itself and its neighbors, the number of its active neighbors, etc. For example, a sensor node is eligible to enter the sleep state if it has one or more active neighbors (see, e.g., [5, 16, 39, 50, 54]).

The neighbors of a node are those nodes that can communicate directly with this node. The communication model of a node determines its neighbors. A commonly used model is the *communication disk model*, which is a disk centered at a sensor with radius the *transmission range* R_c. A direct communication link exists between two nodes if their Euclidean distance is not larger than the transmission range. The transmission range depends on the transmission power. If the transmission power is adjustable, so is the transmission range.

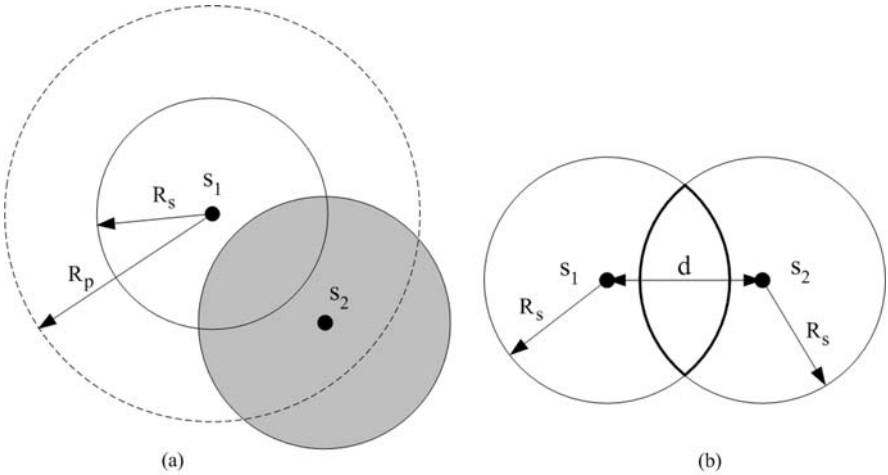

Fig. 7.10 Illustration of neighbor distance based scheduling: (**a**) An active sensor within the probing area; (**b**) The intersection between two nodes' sensing disks

Neighbor Distance-Based Scheduling A simple approach is that a sensor is eligible to enter the sleep state if it finds an active neighbor not far away from itself. For example, in the *Probing Environment and Adaptive Sleeping* (PEAS) scheduling protocol [54], a sensor decides to sleep if it finds at least one active neighbor within its probing area. The probing area of a sensor is a disk centered at the sensor with the radius the probing range, as illustrated by the dashed disk in Fig. 7.10(a). In PEAS, each sensor sleeps for an exponentially distributed duration. When a sensor wakes up, it probes whether there exists any other active sensor within its probing area by sending a probe message. Any active sensor that hears this probe message should reply. If at least one reply is received by this probing sensor, then it enters the sleep state again for another random interval. Otherwise, it enters the active state till its death. Evidently, the probing range controls the density of the active nodes, and the larger the probing range, the less the coverage ratio. In Fig. 7.10(a), the sensor s_1 is eligible for sleep as it has an active neighbor s_2 within its probing area.

Tian and Georganas [39] propose a closest-neighbor-based scheduling algorithm, where a sensor node is eligible to sleep if the distance to its closest active neighbor is less than a predefined threshold. Indeed, if the distance between two nodes is not more than $2R_s$, then there is some intersection area covered by both sensors. As shown in Fig. 7.10(b), the intersection lens is the area covered by both sensor nodes. It is called *auxiliary observable area* of node s_1 offered by node s_2, and vice versa. The area of the intersection lens depends on the distance between the two nodes and can be computed as follows:

$$A(s_i) \cap A(s_j) = \begin{cases} R_s^2 \cos^{-1}(\frac{d}{2R_s}) - \frac{d}{2}\sqrt{4R_s^2 - d^2}, & d \leq 2R_s, \\ 0, & d > 2R_s. \end{cases}$$

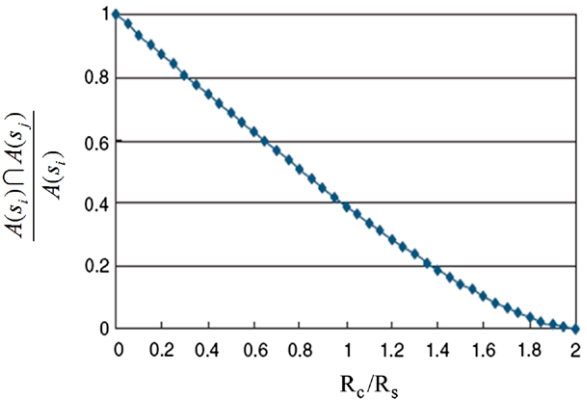

Fig. 7.11 Normalize intersection area of two sensors (reproduced from [39], ©2004, Elsevier)

Figure 7.11 plots the auxiliary observable area normalized with respect to the disk area, which is redrawn from [39]. This result can be used to set a maximal distance $d_{\max}$ such that the uncovered area of a sensing disk by its closest neighbor is less than a predefined threshold. In the process of sensor activity scheduling, a node checks whether it has an active neighbor whose distance to itself is not larger than $d_{\max}$. If so, then this node is eligible to enter into the sleep state.

Neighbor-Number-Based Scheduling Wu et al. [16, 50] analyze the relation between the coverage of a sensor's sensing disk and the number of its *sensing neighbors*. A sensor node is a sensing neighbor of another one if their Euclidean distance is not larger than the sensing range R_s. Let $\mathcal{N}_i$ and $n = |\mathcal{N}_i|$ denote the set and the number of sensing neighbors of a sensor s_i, respectively. They show that if sensor's sensing neighbors cover its sensing disk, i.e., $\bigcup_{s \in \mathcal{N}_i} A(s) \supseteq A(s_i)$, then we can always find a subset of $\mathcal{N}_i$, denoted by $\mathcal{N}_i'$, such that $\bigcup_{s_j \in \mathcal{N}_i'} A(s_j) \supseteq A(s_i)$ and $3 \leq |\mathcal{N}_i'| \leq 5$. This suggests that if a sensor's sensing disk is covered by its sensing neighbors, then we need to choose at least three sensing neighbors and at most five sensing neighbors to cover the sensor's sensing disk. As shown in Fig. 7.12(a), the sensor node s_1 has four sensing neighbors, namely, s_2, s_3, s_4, and s_5. The four sensing neighbors can completely cover the s_1's sensing disk, and actually only three of them, namely, s_2, s_3, and s_4, need to be active to completely cover the s_1's sensing disk.

This result, however, is not very useful in practice, since it is usually not easy to check whether a sensor is completely covered without accurate geography information. Furthermore, it is also possible that the sensing neighbors cannot completely cover the sensing disk, as illustrated by Fig. 7.12(b). On the other hand, it may be more useful to estimate the probability that a sensor's sensing disk is completely covered and its average covered area of the sensing disk. Suppose that there are n sensing neighbors randomly and uniformly distributed within the sensing disk of a sensor node s. Let A_n denote the area within this sensor's sensing disk that is covered by these n sensing neighbors, that is, $A_n = (\bigcup_{j=1}^{n} A(s_j)) \cap A(s)$. Wu et al. [50] provide an upper and a lower bound of the probability that the sensor s's sensing

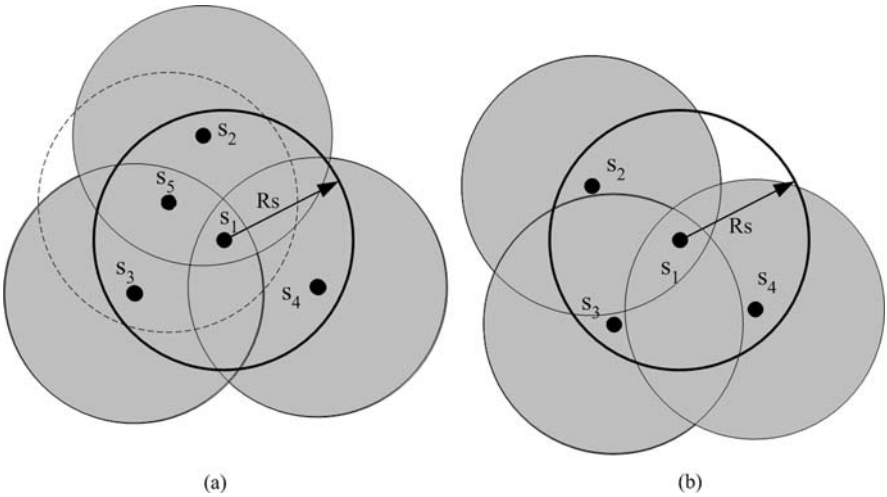

Fig. 7.12 Illustration of the coverage by sensing neighbors: (**a**) The sensing disk is completely covered; (**b**) Some area of the sensing disk is not covered

disk is fully covered by its sensing neighbors, i.e., $\Pr\{A(s) = A_n\}$, as follows:

$$1 - n0.609^{n-1} \leq \Pr\big\{A(s) = A_n\big\} \leq 1 - n0.609^{n-1} + \frac{n(n-1)}{2}(0.276)^{n-1}.$$

They also provide a lower bound for the average covered area $\mathbb{E}[A_n]$ as follows:

$$\mathbb{E}[A_n] \geq \pi R_s^2 \times \left[1 - 0.609^n - \left(\frac{n}{6} - 0.109\right)0.109^{n-1}\right].$$

Given the coverage requirement $\mathbb{E}[A_n]$, a sensor can determine the least number of its active sensing neighbors from the above formula.

Tian and Georganas [39] provide analysis and simulation results for the relation between the coverage of a sensor's sensing disk and the number of its 1-hop neighbors. Let s_j, $j = 1, \ldots, n$, denote the 1-hop neighbor of a sensor s, and let $d(s_j, s) \leq R_c$, where $d(s_j, s)$ is the Euclidean distance between s_j and s, and R_c is the transmission range. They define the auxiliary observable area of a sensor as the union of the intersections between itself and its neighbors, that is, $\bigcup_{j=1}^{n}(A(s_j) \cap A(s))$. They show that the average auxiliary observable area offered by one neighbor is only 25% of the sensor's sensing disk when $R_c = 2R_s$. The calculation of the auxiliary observable area offered by $n > 2$ neighbors is much complicated, and simulations have been used to provide insights on the relation between the coverage of a sensor's sensing disk and the number of its neighbors. Their simulation results are redrawn in Fig. 7.13. If a threshold of minimal auxiliary observable area is chosen a priori for each individual sensor node, then the minimal number of neighbors, $n_{\min}$, can be obtained through simulations. For example, from their simulation results, if the threshold is set as 80%, then at least two neighbors

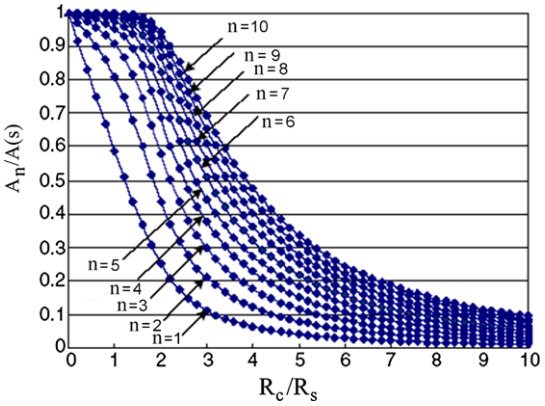

Fig. 7.13 Simulation results of normalized A_n (reproduced from [39], ©2004, Elsevier)

are needed if $R_c = R_s$, and at least six neighbors are needed if $R_c = 2R_s$. In the process of sensor activity scheduling, a node examines whether the number of its active neighbors is equal to or larger than $n_{\min}$. If so, this node can enter the sleep state.

7.3.3 Example Scheduling Protocols

We elaborate two sensor activity scheduling protocols that can provide high area coverage ratio: One is the *Probing Environment and Adaptive Sleeping* (PEAS) scheduling protocol [54], and the other is the *Layered Diffusion-based Coverage Control* (LDCC) [47]. Both protocols are very easy to implement: They do not require nodes' location information, need negligible computations, and incur few message overheads.

Probing Environment and Adaptive Sleeping (PEAS) In PEAS, three states are used, namely, the ACTIVE, SLEEP, and PROBE states. The probe state is a transient state. Figure 7.14 plots the state transition diagram for a node. First, all nodes are in the SLEEP state. Each node sleeps for an exponentially distributed duration generated according to the probability density function (PDF) $f(t_s) = \lambda e^{-\lambda t_s}$, where λ is the *probing rate* of the node, and t_s denote the sleeping time duration.

After a node wakes up, it enters the PROBE state. A probing node seeks to detect whether any active node is present within a certain probing range R_p. Any active node(s) within that range should respond with a REPLY message, also sent within the range of R_p. It is possible that multiple active nodes exist within R_p when a node probes. To reduce collisions, each active node waits for a small random time before it sends back the REPLY message.

If the probing node hears a REPLY message, it goes back to the SLEEP state for another random period of time t_s, generated according to the same PDF. However, λ is adjusted according to an adaptive sleeping algorithm. If the probing node does not hear any REPLY message, it enters the active state.

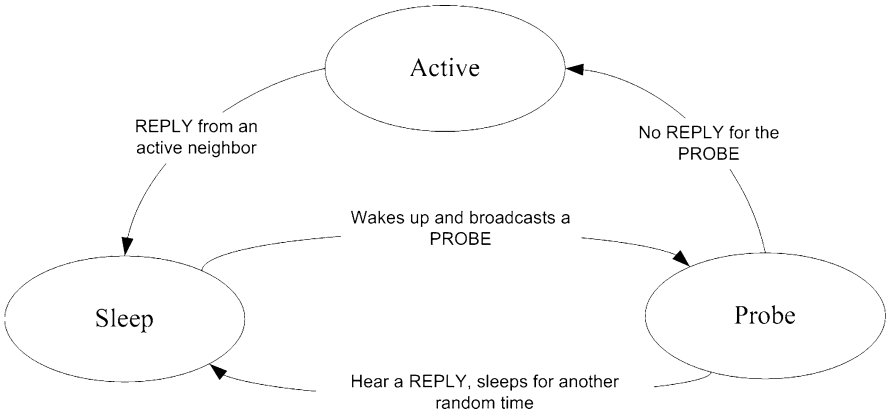

Fig. 7.14 Node state transition diagram in PEAS (reproduced from [54], ©2006, Springer)

The adaptive sleeping algorithm adjusts the probing rate λ of each sleep node so as to keep the aggregate probing rate $\bar{\lambda}$ of all the sleep neighbors of each active node at about a desired rate λ_d, which is specified by application requirements. Each active node maintains a counter N which records how many PROBE messages have been received, and the most recent time t_0 when N is set to 0.

When an active node hears the first PROBE message, it sets the counter to 0, and t_0 to the current time. After that, each time a new PROBE message is received, the counter increases by one. Eventually, when the counter reaches a threshold N_{th}, a measurement $\hat{\lambda}$ is calculated as

$$\hat{\lambda} = \frac{N_{\text{th}}}{t - t_0},$$

where t is the current time. This node then sets t_0 to t, resets the counter to 0, and repeats the above process. Whenever an active node receives a PROBE message, it includes its current probing measurement $\hat{\lambda}$ in its REPLY message.

Upon receiving a REPLY message from an active node, a probing node updates its current probing rate λ as

$$\lambda^{\text{new}} = \lambda \frac{\lambda_d}{\hat{\lambda}}.$$

Then the probing node will use λ^{new} to generate a new sleeping period t_s according to the probability density function $f(t_s) = \lambda^{\text{new}} e^{-\lambda^{\text{new}} t_s}$.

Layered Diffusion-based Coverage Control (LDCC) In LDCC, two states are used, namely, the ACTIVE and SLEEP states. The time line is divided into consecutive rounds of equal lengths. All nodes are in the active state at the beginning of each round, and a process called *layered diffusion* is executed to deactivate some active nodes. After the layered diffusion process, some sensor nodes enter the sleep state and return back to the active state at the beginning of next round; then, the layered diffusion is repeated again.

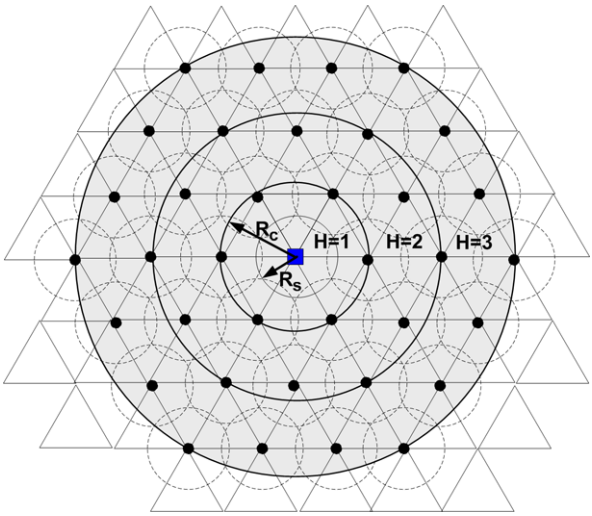

Fig. 7.15 Illustration of tessellation by equilateral triangles

Basic Idea The basic idea of the LDCC protocol is to dynamically emulate a triangle tessellation via a layered diffusion process. As shown in Fig. 7.15, a triangle tessellation consists of equilateral triangles and can completely cover a plane. It is well known that putting disks centered at the vertices of such a triangle tessellation with each triangle's side length $\sqrt{3}R_s$ requires the least number of disks to provide complete area coverage [29]. Therefore, we can choose those nodes with locations best approximating the triangular tessellation to be active. This is also the basic idea behind the OGDC protocol [57], which sequentially activates sensor nodes to emulate a triangle tessellation. However, the approach in OGDC requires the exact nodes' locations to emulate a triangle tessellation.

Instead of using nodes' exact location information, the LDCC protocol applies hop count information in the process of emulating a triangular tessellation. Given that all sensor nodes use a fixed transmission power, the hop count measures how many transmissions are needed for a sensor node to deliver its packets to the base station. Indeed, the hop count information provides a kind of geometric division of the sensor field. For example, in Fig. 7.16, given the infinite node density, the sensor nodes on the circle with radius R_c are those nodes with hop count $H = 1$; those nodes within the ring in between the two circles with radii R_c and $2R_c$ are those nodes with hop count $H = 2$; and so on. Note that the radius of each ring may not be an exact multiple of the communication range in a randomly deployed sensor network with finite density. Observing the triangular tessellation shown in Fig. 7.15, if the transmission range is set appropriately, the triangular tessellation can proceed by using one sensor node with hop count H and the other two nodes with hop count $H + 1$.

We further explain the basic idea of the LDCC protocol using Fig. 7.16. In LDCC, a hop count originator is a node or the sink that sets up the hop counts for all other nodes. The hop count information can be obtained via controlled flooding originated from the hop count originator. In Fig. 7.16, there are three rings, each

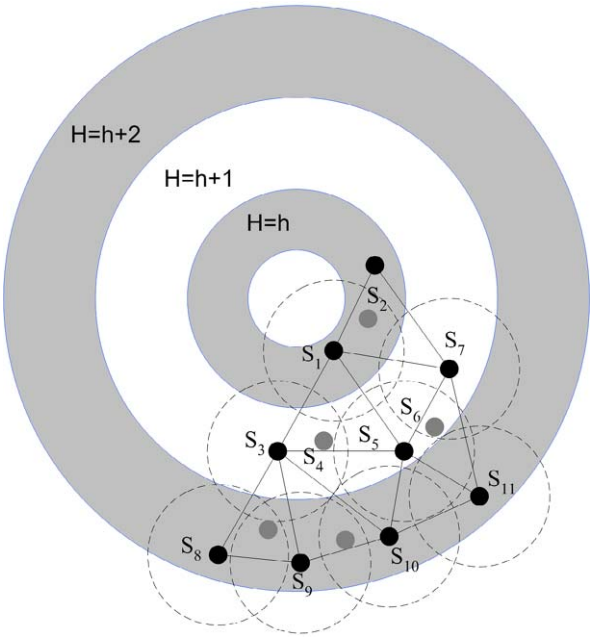

Fig. 7.16 Illustration of LDCC by an example of how to select active sensors

of which consists of nodes with hop count $H = h, h + 1$, and $h + 2$, respectively. At first, all sensor nodes are set to the active state and listen to their neighbors. We assume that each sensor node has a timer that can be set or reset to different time periods. Whenever the timer expires, a sensor node sets itself to the active state and sends out the ACTIVE message. The process of coverage control normally starts when a sensor node with a small hop count sends out an ACTIVE message to its neighbors upon expiration of its timer. For example, suppose that sensor node s_1 first sends out an ACTIVE message with its hop count information attached. After receiving an ACTIVE message, each node either needs to set its state to sleep or needs to reset its timer to a different time period according to its hop count information (the number of received ACTIVE messages and the hop count information contained in the received ACTIVE message). A rule of thumb is that given that the transmission range R_c is appropriately set, if a sensor node receives two or more ACTIVE messages sent from its neighbors with the same hop count as itself, it can then set itself to sleep.

In Fig. 7.16, after sensor nodes s_3, s_4, s_5, s_6, s_7 receive the ACTIVE message from sensor node s_1 as the first ACTIVE message ever received, each of them resets its timer to a randomly selected value between 1 and a maximum number. Suppose that the timer of sensor node s_3 expires first; then it sets itself to an active state and sends out an ACTIVE message that can be received by nodes s_4 and s_5. Again, nodes s_4 and s_5 reset their respective timer. Suppose that the timer of node s_5 expires earlier than that of node s_4, and node s_5 sends out an ACTIVE message that can also be received by node s_4. Then, after receiving two ACTIVE messages from node s_3 and s_5 with the same hop count as itself, node s_4 sets itself to a sleep state.

Furthermore, to enable the rotation between the sleep and the active state for each sensor node, the ACTIVE message can also attach a time period value stating how long its active state is, which will also be used for a sleep node to set its sleep time period. Therefore, after some interval, all nodes become active again, and the above procedure repeats for selecting the active sensor nodes for the next round.

Protocol Description The term *layered diffusion* is used to describe a message exchange process in which messages of some particular types are likely to be first generated by a hop count originator and then diffused from the nodes in a ring closest to the originator to the nodes in a ring farthest from the originator. That is, for some particular message types, a sensor node with a small hop count ignores the messages sent out by another node with a larger hop count.

The execution of the LDCC protocol is divided into rounds, and in each round, the originator initiates the coverage control process by sending out an ACTIVE message. An ACTIVE message indicates that the sender has set itself to an active state and includes the sender's hop count (H^r), transmission power (P^r), and active time in this round (T^r). Note that the ACTIVE message sent by the originator has hop count 0. Each sensor node maintains a Boolean variable (B) as a flag to indicate that a new round of coverage control has started. B is initialized to *true* when the sensor network is deployed. Each sensor node also maintains several counters N_{tm}, N_{ms}, and N_{me}. N_{tm} records the time passed after a node receives the last ACTIVE message; N_{ms} and N_{me} are to record the number of received ACTIVE messages sent from the nodes with hop count smaller than its own and from the nodes with the same hop count as its own, respectively. These counters are initialized to zero at the time when the sensor network is deployed.

The internal logic of the LDCC protocol is implemented with using the above flag and counters to set a timer with a different expiration period to control the sensor node state upon timer expiration. We assume that all sensor nodes have the same timer tick unit. Furthermore, let N_0 denote the time for transmitting and receiving a message (transmission delay) plus the time for processing a message (processing delay), which is assumed to be the same across different sensor nodes. The timer is initialized to a very large value $T_{\max}$ that is greater than or equal to the sensor node lifetime at the time when the sensor network is deployed. When we say that a sensor node sets its timer to T, this means that if the current time instant (or tick) is t, then the timer expires, and some event will be triggered at the time instant (or tick) $t + T$.

Figure 7.17 shows the flowchart of how a sensor node operates at each tick, where the coverage control part is used to decide the sensor node state (*active* or *sleep*) and the time period of the state for each round. After deciding on the state in a new round, the sensor node will stay in that state. When the timer for starting a new round expires, it changes its state to *active* no matter which state it is in the last round. It clears the counters for the coverage control.

The flowchart for the coverage control is given by Fig. 7.18. After receiving an ACTIVE message, a sensor node compares its own hop count H with the hop count contained in the received ACTIVE message H^r. If $H^r < H$, then it sets two counters N_{tm} and N_{ms}, records the transmission power (P^r) and the time duration of the

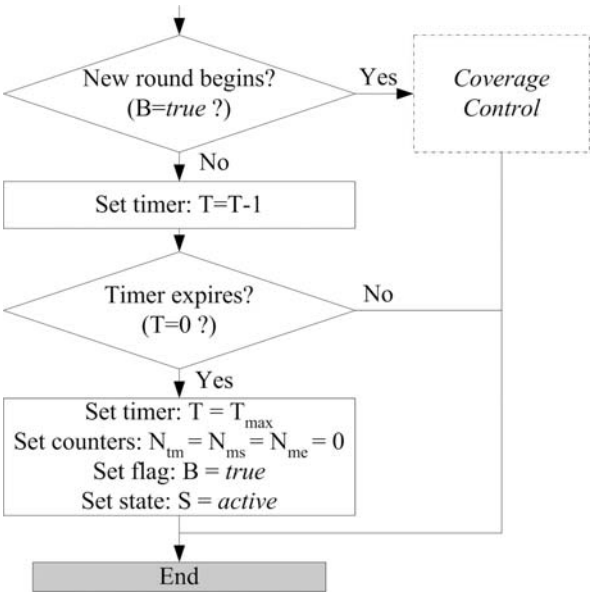

Fig. 7.17 Node operation flowchart at each tick in LDCC

active state (T^r) of the sender, and sets a timer with a randomly selected expiration time. We use $T \sim (T_a, T_b)$ to denote that the value of T is selected randomly from a uniform distribution ranging from T_a to T_b , where T_a, T_b, and T_c denote different thresholds, and $T_a \geq N_0$ and $T_b, T_c > T_a$. If $H^r = H$, then it increases the counter N_{me} by one and records the transmission power (P^r) and the time duration of the *active* state (T^r) of the sender. If it receives two ACTIVE messages from nodes with the same hop count ($N_{\text{me}} = 2$), then it enters the *sleep* state. Otherwise, it resets its timer and the counter N_{tm}.

Note that as a feature of layered diffusion, a node ignores all ACTIVE messages sent from nodes with a hop count greater than its own. The counter N_{tm} records the time period from when it receives the most recently ACTIVE message either from the node with $H^r = H$ or from the node with $H^r < H$ to when it changes its state. After a node decides its state (as either *active* or *sleep*), it sets its state duration as $T^r - N_{\text{tm}} - N_0$. Note that T^r always records the time duration of the active state from the most recently received ACTIVE message. Therefore, by the propagation of T^r and the use of N_{tm}, all *active* and *sleep* nodes are expected to transition back to the active state at almost the same time, given that no time drift occurs in an individual node.

7.3.4 Notes and Comments

The network lifetime also depends on how energy is consumed for transmitting and receiving sensing data. In the multi-hop network scenario where all active sensors

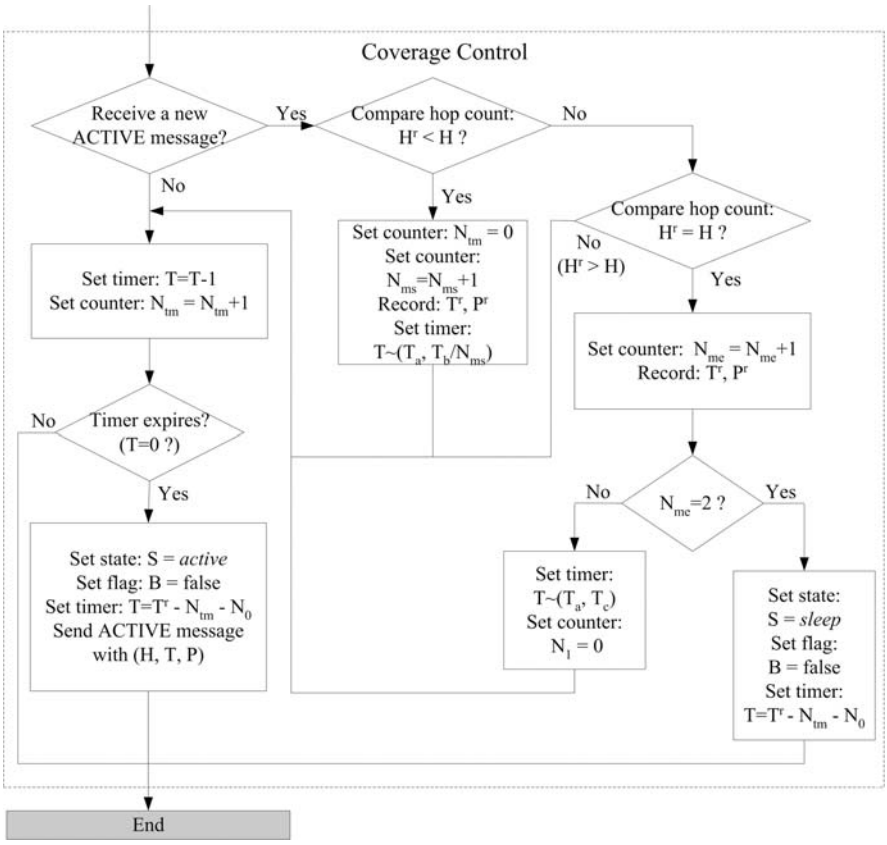

Fig. 7.18 Node operation flowchart for coverage control in LDCC

need to transmit their sensing data to the sink, the nodes closer to the sink will have to consume more energy for relaying data than those nodes farther away from the sink. In such a case, we need to balance energy consumption for nodes with different distances to the sink [10, 11, 53]. For example, one approach is to activate fewer nodes farther away from the sink. Deng et al. [10, 11] propose to extend the Random Independent Sleeping (RIS) scheme so that nodes farther away from the sink have smaller probabilities to be self-activated.

The partial area coverage problem discussed so far are all based on the simple sensing disk coverage model. In what follows, we list some other problem variants based on different assumptions, objectives, or other coverage models. The problem of sensor activity scheduling for preserving area coverage has also been studied under other coverage models, such as directional coverage models [37], disk coverage models with adjustable sensing ranges [49], detection coverage models [2, 14, 21, 36], and estimation coverage models [43, 46].

7.4 Preserving Area Coverage and Network Connectivity

The communication unit in a sensor node (e.g., the radio transceiver in a wireless sensor node) is in general independent of the sensor unit of the sensor [3]. A sensor activity scheduling algorithm that is designed to schedule the state (sleep or active) of nodes' sensor units does not necessarily imply that an inactive sensor node shall also shut off its communication unit and cannot serve as an intermediate relay for forwarding data. In some cases, it might be desirable for a node with inactive sensor unit to turn off its communication unit as well. However, network connectivity may not be guaranteed if only those active sensor nodes that are selected by a sensor activity scheduling algorithm are used for data transport. In these cases, sensor activity scheduling should ensure both area coverage and network connectivity.

7.4.1 Relation Between Area Coverage and Network Connectivity

Let us assume that the sensor coverage model is a disk with radius R_s (sensing range), and the node communication model is also a disk with radius R_c (communication range). The following theorem states the relation between complete area coverage and the network connectivity with respect to the relation between R_s and R_c under the assumptions of the sensor disk model and communication disk model.

Theorem 7.1 (Xing et al. [51], Theorem 1, Zhang and Hou [57], Lemma 1) *Suppose that a set of sensors $\mathcal{S}$ provides complete* 1*-coverage for a convex field. Then, if $R_c \geq 2R_s$, these sensors form a* 1*-connected network.*

Xing et al. [51] apply the Voronoi diagram in proving the statement. We refer the reader to Appendix A for a brief introduction about the Voronoi diagram. Figure 7.19 shows the Voronoi diagram of a set of sensor nodes completely covering a convex field. Let $\text{Vor}(u)$ denote the Voronoi cell of a node u. We first prove that any two nodes whose Voronoi cells are adjacent can communicate with each other. As illustrated in Fig. 7.19, p is the Voronoi vertex of three adjacent Voronoi cells $\text{Vor}(u)$, $\text{Vor}(v)$, and $\text{Vor}(w)$. According to the definition of a Voronoi diagram, u, v, and w are equally distant from p and are closest to p among all nodes. Hence p must be covered by u, v, and w; otherwise, it will not be covered by any nodes. According to the triangle inequality, we have

$$|uv| \leq |pu| + |pv| < 2R_s \leq R_c.$$

The network is connected if there is a communication path between any two nodes s and t in the network. When constructing Voronoi diagram, every node has at least one neighboring Voronoi cell. We can thus construct a line segment $\overline{st}$ that intersects consecutive Voronoi cells $\text{Vor}(s) = \text{Vor}(n_1)$, $\text{Vor}(n_2), \ldots, \text{Vor}(n_k) = \text{Vor}(t)$. Any two consecutive nodes in the series n_1 to n_k, since their Voronoi cells are adjacent,

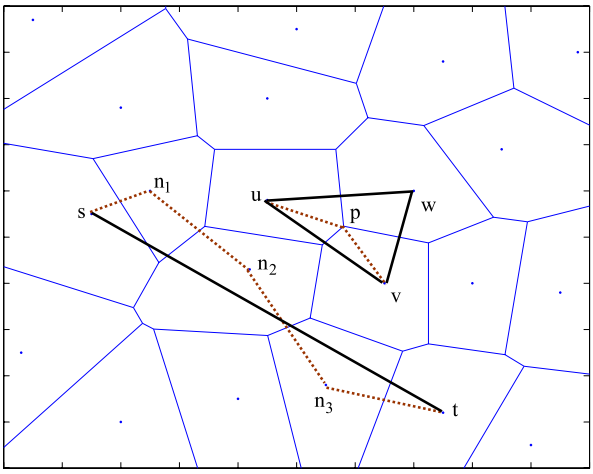

Fig. 7.19 The Voronoi diagram of sensor nodes completely covering a convex field

can communicate with each other. Hence nodes n_1 and n_k constitute a communication path from s to t. The dotted path between s and t in Fig. 7.19 illustrates such a path.

This theorem indicates that in case of $R_c \geq 2R_s$, a sensor activity scheduling algorithm that provides complete area coverage can also guarantee network connectivity. Tian and Georganas [40] extend this result to a more general scenario where the complete area coverage is relaxed to the union of coverage area of a set of sensors $\mathcal{S}$.

Theorem 7.2 (Tian and Georganas [40]) *Suppose that a set of sensors $\mathcal{S}$ can form a connected network and can cover an area $A \doteq \bigcup_{s\in\mathcal{S}} A(s)$. If a subset of sensors $\mathcal{S}_a \subseteq \mathcal{S}$ can also preserve the same coverage, $A_a \doteq \bigcup_{s\in\mathcal{S}_a} A(s) = A$, and $R_c \geq 2R_s$, then the sensors in $\mathcal{S}_a$ also form a connected network.*

The difference between the two theorems is that although all sensor nodes in $\mathcal{S}$ form a connected network, they may not be able to provide complete area coverage. The above two theorems can also be extended to k-coverage and higher-degree connectivity. A graph is said k-connected if there does not exist a set of $k-1$ vertices whose removal disconnects the graph. If a set of sensors $\mathcal{S}$ provides complete k-coverage for a convex field $\mathcal{A}$, then, if $R_c \geq 2R_s$, these sensors form a k-connected communication graph.

7.4.2 Connected Coverage Scheduling

The objectives of a connected coverage scheduling algorithm are to select active sensor nodes such that the area coverage requirement can be satisfied by these ac-

tive sensors and these selected nodes (together with the sink) also form a connected network. As discussed in the previous subsection, if $R_c \geq 2R_s$, a sensor activity scheduling algorithm preserving complete area coverage can also guarantee network connectivity. In the case of $R_c < 2R_s$, such a scheduling algorithm needs to be modified: Some extra nodes may need to be activated (or some nodes cannot be deactivated) in order to ensure complete area coverage and maintain network connectivity. For example, in case of $R_c < 2R_s$, the self inactivation rule in the OGDC is modified as follows. A node can sleep if (1) its coverage area is completely covered by its active neighbors, and (2) its active neighbors are all connected without it. Obviously, the second rule is added in order to maintain network connectivity. In what follows, we introduce two typical approaches to ensure both area coverage and network connectivity.

Activate Extra Nodes In this approach, a sensor activity scheduling algorithm is first executed to select sensing active nodes. Then the network connectivity of these active nodes is examined. If these nodes cannot form a connected network, some extra nodes are activated [32, 41].

For example, Liu et al. [32] propose an *extra-on* rule to select more active nodes to guarantee that each active node can find a path with the minimum hops to the sink. First, a synchronous decision approach of RIS is applied to generate k disjoint sensor set covers. That is, each sensor node randomly selects a number i drawn from a uniform distribution from 1 to k and activates itself in the corresponding round. Then a controlled flooding is enforced to setup the hop count relative to the sink for every node. A node s_1 is called an upstream node of another node s_2 if they are 1-hop neighbors and their hop counts satisfy $h_1 = h_2 - 1$; s_2 is also called the downstream node of node s_1. The extra-on rule is as follows. If a node s_1 is active in a round i but none of its upstream nodes are active in the same round, then s_1 should activate at least one of its upstream node in the round i.

Select Connected Nodes In this approach, the network connectivity constraint is embedded in the sequential selection of active nodes. At each selection step, only the nodes connected to at least one of the already selected nodes are eligible to be activated [13, 18, 19, 30, 63].

Let $\mathcal{S}_a$ be the set of selected sensor nodes, and let $\mathcal{N}(\mathcal{S}_a)$ denote the set of sensors that are neighbors of at least one sensor node in $\mathcal{S}_a$ but not in $\mathcal{S}_a$. A greedy algorithm to select connected active nodes can be as follows [18, 19]. At each stage of selecting next active nodes, only those nodes in $\mathcal{N}(\mathcal{S}_a)$ are considered, and the node in $\mathcal{N}(\mathcal{S}_a)$ that can maximize a profit function is selected. An example profit function can be the area of the region that can be covered by this sensor node but not covered by those already selected sensor nodes. The greedy algorithm stops if the area coverage requirement is satisfied. Since only nodes in $\mathcal{N}(\mathcal{S}_a)$ is considered at each stage, network connectivity can be guaranteed. Obviously, this is a variant of the greedy set cover heuristic for the classic set covering problem. By using different profit functions, different objectives, such as minimal energy consumption and optimal balance between energy consumption and coverage requirement, can also be achieved.

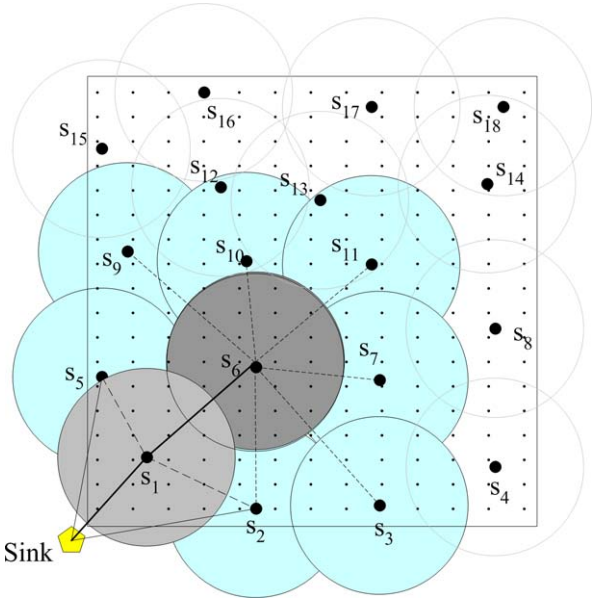

Fig. 7.20 Illustration of sequentially selecting connected nodes for area coverage

An example is illustrated in Fig. 7.20. At first, $\mathcal{S}_a = \emptyset$, and $\mathcal{N}(\mathcal{S}_a) = \{s_1, s_2, s_5\}$. The node s_1 is first selected as it covers the most uncovered area. Then $\mathcal{S}_a = \{s_1\}$ and $\mathcal{N}(\mathcal{S}_a) = \{s_2, s_5, s_6\}$. At the next selection, the node s_6 is selected, and $\mathcal{S}_a$ and $\mathcal{N}(\mathcal{S}_a)$ are updated. This process terminates if the whole sensor field is covered.

7.4.3 Notes and Comments

In the greedy algorithm of selecting connected nodes, the profit (or cost) function of a sensor node can also take into consideration of the node's residual energy and transmission energy consumption [45, 59]. There are also some other extensions and variants of joint sensor activity scheduling and transmission route selection in the literature. For example, under the disk coverage model with adjustable sensing ranges, Zhou et al. [60–62] extend the greedy algorithm for selecting connected nodes to achieve connected k-coverage. Ammari and Das [4] propose a joint k-coverage and hybrid forwarding protocol for three-dimensional networks.

References

1. Abrams, Z., Goel, A., Plotkin, S.: Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In: ACM International Symposium on Information Processing in Sensor Networks (IPSN), pp. 424–432 (2004)
2. Ahmed, N., Kanhere, S.S., Jha, S.: Probabilistic coverage in wireless sensor networks. In: IEEE Conference on Local Computer Networks (LCN), pp. 672–681 (2005)

3. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: A survey. Computer Networks **39**(4), 393–422 (2002)
4. Ammari, H.M., Das, S.K.: Joint k-coverage and hybrid forwarding in duty-cycled three-dimensional wireless sensor networks. In: IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), pp. 170–178 (2008)
5. Bai, H., Chen, X., Li, B., Han, D.: A location-free algorithm of energy-efficient connected coverage for high density wireless sensor networks. Discrete Event Dynamic Systems (Springer) **17**(1), 1–21 (2007)
6. Boukerche, A., Fei, X.: A Voronoi approach for coverage protocols in wireless sensor networks. In: IEEE Global Telecommunications Conference (Globecom), pp. 5190–5194 (2007)
7. Boukerche, A., Fei, X., Araujo, R.B.: An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange. Computer Communications (Elsevier) **30**(14–15), 2708–2720 (2007)
8. Cărbunar, B., Grama, A., Vitek, J., Carbunar, O.: Redundancy and coverage detection in sensor networks. ACM Transactions on Sensor Network (ToSN) **2**(1), 94–128 (2006)
9. Choi, W., Das, S.K.: Coverage-adaptive random sensor scheduling for application-aware data gathering in wireless sensor networks. Computer Communications (Elsevier) **29**(17–18), 3476–3482 (2006)
10. Deng, J., Han, Y.S., Heinzelman, W.B., Varshney, P.K.: Balanced-energy sleep scheduling scheme for high-density cluster-based sensor networks. Computer Communications **28**(14), 1631–1642 (2005)
11. Deng, J., Han, Y.S., Heinzelman, W.B., Varshney, P.K.: Scheduling sleeping nodes in high density cluster-based sensor networks. ACM/Kluwer Mobile Networks and Applications (MONET) **10**(6), 825–835 (2005)
12. Du, X., Lin, F.: Maintaining differentiated coverage in heterogeneous sensor networks. EURASIP Journal on Wireless Communications and Networking **2005**(4), 565–572 (2005)
13. Funke, S., Kesselman, A., Kuhn, F., Lotker, Z., Segal, M.: Improved approximation algorithms for connected sensor cover. Wireless Networks (Springer) **13**(2), 153–164 (2007)
14. Fusco, G., Gupta, H.: Selection and orientation of directional sensors for coverage maximization. In: IEEE Communications Society 6th Annual Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), pp. 1–9 (2009)
15. Gallais, A., Carle, J., Simplot-Ryl, D., Stojmenovic, I.: Localized sensor area coverage with low communication overhead. IEEE Transactions on Mobile Computing **7**(5), 661–672 (2008)
16. Gao, Y., Wu, K., Li, F.: Analysis on the redundancy of wireless sensor networks. In: ACM International Conference on Wireless Sensor Networks and Applications (WSNA), pp. 108–114 (2003)
17. Gui, C., Mohapatra, P.: Power conservation and quality of surveillance in target tracking sensor networks. In: ACM International Conference on Mobile Computing and Networking (MobiCom), pp. 129–143 (2004)
18. Gupta, H., Das, S.R., Gu, Q.: Connected sensor cover: Self-organization of sensor networks for efficient query execution. In: ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 189–200 (2003)
19. Gupta, H., Zhou, Z., Das, S.R., Gu, Q.: Connected sensor cover: Self-organization of sensor networks for efficient query execution. IEEE/ACM Transactions on Networking **14**(1), 55–67 (2006)
20. Hall, P.: Introduction to the Theory of Coverage Processes. Wiley, New York (1988)
21. Hefeeda, M., Ahmadi, H.: A probabilistic coverage protocol for wireless sensor networks. In: IEEE International Conference on Network Protocols (ICNP), pp. 1–10 (2007)
22. Hsin, C., Liu, M.: Randomly duty-cycled wireless sensor networks: Dynamics of coverage. IEEE Transactions on Wireless Communications **5**(11), 3182–3192 (2006)
23. Hua, C., Yum, T.S.P.: Asynchronous random sleeping for sensor networks. ACM Transactions on Sensor Networks (ToSN) **3**(3), 1–25 (2007)
24. Huang, C.F., Tseng, Y.C.: The coverage problem in wireless sensor networks. Mobile Networks and Applications **10**(4), 519–528 (2005)

25. Huang, C.F., Tseng, Y.C., Lo, L.C.: The coverage problem in three-dimensional wireless sensor networks. In: IEEE Global Telecommunications Conference (Globecom), pp. 3182–3186 (2004)
26. Huang, C.F., Lo, L.C., Tseng, Y.C., Chen, W.T.: Decentralized energy-conserving and coverage-preserving protocols for wireless sensor networks. In: International Symposium on Circuits and Systems (ISCAS) (2005)
27. Huang, C.F., Tseng, Y.C., Wu, H.L.: Distributed protocols for ensuring both coverage and connectivity of a wireless sensor network. ACM Transactions on Sensor network (ToSN) **3**(1), 1–24 (2007)
28. Jiang, J., Dou, W.: A coverage-preserving density control algorithm for wireless sensor networks. In: The 3rd International Conference on Ad-Hoc Networks and Wireless (Adhoc-Now), also in LNCS, vol. 3158, pp. 42–55 (2004)
29. Kershner, R.: The number of circles covering a set. American Journal of Mathematics **61**(3), 665–671 (1939)
30. Liu, Y., Liang, W.: Approximate coverage in wireless sensor networks. In: Local Computer Networks (LCN), pp. 68–75 (2005)
31. Liu, B., Towsley, D.: A study of the coverage of large-scale sensor networks. In: IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), pp. 475–483 (2004)
32. Liu, C., Wu, K., Xiao, Y., Sun, B.: Random coverage with guaranteed connectivity: Joint scheduling for wireless sensor networks. IEEE Transactions on Parallel and Distributed Systems **17**(6), 562–575 (2006)
33. Lu, J., Wang, J., Suda, T.: Scalable coverage maintenance for dense wireless sensor networks. EURASIP Journal of Wireless Communications and Networking (2007)
34. Noh, Y., Lee, S., Kim, K.: Central angle decision algorithm in coverage-preserving scheme for wireless sensor networks. In: IEEE Wireless Communications and Networking Conference (WCNC), pp. 2492–2496 (2008)
35. Quang, V.T., Miyoshi, T.: An algorithm for sensing coverage problem in wireless sensor networks. In: IEEE Sarnoff Symposium, pp. 1–5 (2008)
36. Ren, S., Li, Q., Wang, H., Chen, X., Zhang, X.: Design and analysis of sensing scheduling algorithms under partial coverage for object detection in sensor networks. IEEE Transactions on Parallel and Distributed Systems **18**(3), 334–350 (2007)
37. Tezcan, N., Wang, W.: Self-orienting wireless multimedia sensor networks for maximizing multimedia coverage. In: IEEE International Conference on Communications (ICC), pp. 2206–2210 (2008)
38. Tian, D., Georganas, N.D.: A node scheduling scheme for energy conservation in large wireless sensor networks. Journal of Wireless Communications and Mobile Computing **3**(2), 271–290 (2003)
39. Tian, D., Georganas, N.D.: Location and calculation-free node-scheduling schemes in large wireless sensor networks. Ad Hoc Networks (Elsevier) **2**(1), 65–85 (2004)
40. Tian, D., Georganas, N.D.: Connectivity maintenance and coverage preservation in wireless sensor networks. Ad Hoc Networks **3**(6), 744–776 (2005)
41. Tian, Y., Zhang, S.F., Wang, Y.: A distributed protocol for ensuring both probabilistic coverage and connectivity of high density wireless sensor networks. In: IEEE Wireless Communications and Networking Conference (WCNC), pp. 2069–2074 (2008)
42. Tran-Quang, V., Miyoshi, T.: A novel gossip-based sensing coverage algorithm for dense wireless sensor networks. Computer Networks (Elsevier) **53**(13), 2275–2287 (2009)
43. Vashistha, S., Azad, A.P., Chockalingam, A.: Energy efficient area monitoring using information coverage in wireless sensor networks. In: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1–10 (2007)
44. Wang, L., Kulkarni, S.S.: Sacrificing a little coverage can substantially increase network lifetime. Ad Hoc Networks (Elsevier) **6**(8), 1281–1300 (2008)
45. Wang, B., Vikram, S., Chua, K.C., Wang, W.: Information coverage and network lifetime in energy constrained wireless sensor networks. In: IEEE the 32rd Conference on Local Computer Networks (LCN), pp. 512–519 (2007)

46. Wang, B., Chua, K.C., Srinivasan, V.: Connected sensor cover for area information coverage in wireless sensor networks. Journal of Communications System (Wiley) **21**(11), 1181–1203 (2008)
47. Wang, B., Fu, C., Lim, H.B.: Layered diffusion based coverage control in wireless sensor networks. Computer Networks (Elsevier) **53**(7), 1114–1124 (2009)
48. Wu, T.T., Ssu, K.F.: Determining active sensor nodes for complete coverage without location information. International Journal of Ad Hoc and Ubiquitous Computing **1**(1), 38–46 (2005)
49. Wu, J., Yang, S.: Energy-efficient node scheduling models in sensor networks with adjustable ranges. International Journal of Foundations of Computer Science **16**(1), 3–17 (2005)
50. Wu, K., Gao, Y., Li, F., Xiao, Y.: Lightweight deployment-aware scheduling for wireless sensor networks. Mobile Networks and Applications (MNA) (Springer) **10**(6), 837–852 (2005)
51. Xing, G., Wang, X., Zhang, Y., Lu, C., Pless, R., Gill, C.: Integrated coverage and connectivity configuration for energy conservation in sensor networks. ACM Transactions on Sensor Networks **1**(1), 36–72 (2005)
52. Yan, T., Gu, Y., He, T., Stankovic, J.A.: Design and optimization of distributed sensing coverage in wireless sensor networks. ACM Transactions on Embedded Computing Systems **7**(3), 1–40 (2008)
53. Yardibi, T., Karasan, E.: A distributed activity scheduling algorithm for wireless sensor networks with partial coverage. Wireless Networks (Springer), pp. 1–13 (2009). doi:10.1007/s11276-008-0125-2
54. Ye, F., Zhang, H., Lu, S., Zhang, L., Hou, J.: A randomized energy-conservation protocol for resilient sensor networks. Wireless Networks **12**(5), 637–652 (2006)
55. Younis, O., Krunz, M., Ramasubramanian, S.: Location-unaware coverage in wireless sensor networks. Ad Hoc Networks (Elsevier) **6**(7), 1078–1097 (2008)
56. Zhang, H., Hou, J.: On deriving the upper bound of α-lifetime for large sensor networks. In: ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 121–132 (2004)
57. Zhang, H., Hou, J.C.: Maintaining sensing coverage and connectivity in large sensor networks. Journal of Ad Hoc and Sensor Wireless Networks **1**(1–2), 89–124 (2005)
58. Zhang, M., Chan, M.C., Ananda, A.L.: Coverage protocol for wireless sensor networks using distance estimates. In: IEEE the 4th Annual Communications Society on Sensor and Ad Hoc Communications and Networks (SECON), pp. 1–10 (2007)
59. Zhao, T., Zhao, Q.: Lifetime maximization based on coverage and connectivity in wireless sensor networks. Journal of Signal Processing Systems (Springer) (2009). doi:10.1007/s11265-008-0324-1
60. Zhou, Z., Das, S., Gupta, H.: Variable radii connected sensor cover in sensor networks. In: IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON), pp. 387–396 (2004)
61. Zhou, Z., Das, S., Gupta, H.: Fault tolerant connected sensor cover with variable sensing and transmission ranges. In: IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON) (2005)
62. Zhou, Z., Das, S.R., Gupta, H.: Variable radii connected sensor cover in sensor networks. ACM Transactions on Sensor Network (ToSN) **5**(1), 1–36 (2009)
63. Zou, Y., Chakrabarty, K.: A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks. IEEE Transactions on Computers **54**(8), 978–991 (2005)

Chapter 8
Node Movement Strategy

Abstract In one random deployment, it is often assumed that the number of scattered sensors are more than that required by the critical sensor density. Otherwise, complete area coverage may not be guaranteed in this deployment, and some coverage holes may exist. Besides using more sensors to improve coverage, *mobile sensor nodes* can be used to improve network coverage. Mobile sensor nodes are equipped with mobile platforms and can move around after initial deployment. Although a mobile sensor node normally is more expensive than its stationary compeer, it can serve many functionalities and improve network coverage. The design of a node movement strategy usually needs to address the following question:

Where to move and how to efficiently move mobile modes so that area coverage can be optimized?

In different network scenarios, the objectives of nodes' movement are different. In a hybrid network consisting of both stationary and mobile sensor nodes, the objective is mainly to relocate mobile nodes to heal the coverage holes caused by the stationary nodes. In a mobile network consisting of only mobile nodes, the primary objective is to maximize the coverage of these mobile nodes, and in event monitoring scenario, an important objective is to dispatch mobile nodes to the sources of events for better event coverage.

8.1 Healing Coverage Hole

Mobile sensor nodes (for example, Robomote [16] and iMouse [17]) are equipped with mobile platforms and can move around after initial deployment. Coverage holes may exit if the number of deployed nodes are not large enough in one random deployment, or some nodes have run out of energy. In a hybrid network consisting of both stationary and mobile sensor nodes, one objective for using mobile sensor nodes is to minimize coverage holes created by those stationary nodes [7, 14, 23]. When designing a hole healing algorithm, the following issues need to be addressed. First, how to decide the existence of a coverage hole and how to estimate the size of a hole. Second, where are the best target locations to relocate mobile nodes to repair

B. Wang, *Coverage Control in Sensor Networks,*
Computer Communications and Networks,
DOI 10.1007/978-1-84996-059-5_8, © Springer-Verlag London Limited 2010

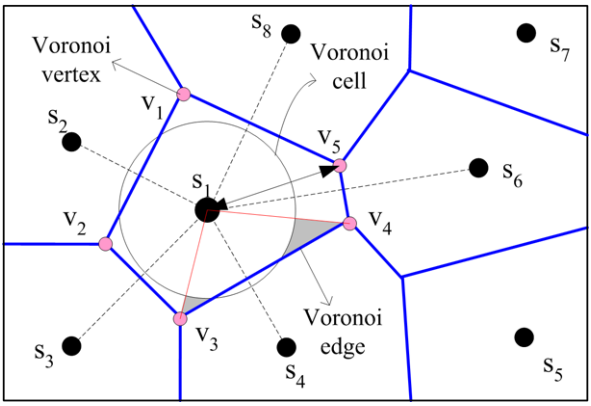

Fig. 8.1 Illustration of using Voronoi diagram to detect and decide a coverage hole

coverage holes. Third, how to dispatch mobile nodes to the target locations while minimizing the moving and messaging cost.

Hole Detection and Size Estimation Voronoi diagram can be used to detect a coverage hole and calculate its size [7, 23]. A Voronoi diagram is first constructed for all stationary sensor nodes, assuming that each node knows its own and its neighbors' coordinates. Wang et al. [23] propose a distributed Voronoi diagram construction algorithm where each node constructs its own Voronoi cell by only considering its 1-hop neighbors. After the Voronoi diagram construction, the sensor field is divided into subregions of Voronoi cells, and each stationary node is within a Voronoi cell. A node is a Voronoi neighbor of another one if they share a Voronoi edge. Figure 8.1 illustrates a Voronoi diagram in a bounded sensor field, where the boundaries of the sensor field also contribute to a Voronoi cell. According to the property of a Voronoi diagram, all the points within a Voronoi cell are closest to only one node that lies within this cell. Therefore, if some points of a Voronoi cell are not covered by its generating node, these points will not be covered by any other sensors and contribute to coverage holes. If a sensor covers all of its Voronoi cell's vertices, then there are no uncovered points within its Voronoi cell; otherwise, uncovered points exist within its Voronoi cell.

Ghosh [7] describes how to compute the uncovered area within a Voronoi cell. They call a triangle consisting of a node and its two adjacent Voronoi vertices a Voronoi triangle. For example, the Voronoi triangle $\triangle s_1 v_3 v_4$ in Fig. 8.1. The line $\overline{v_3 v_4}$ is the perpendicular bisector of the line $\overline{s_1 s_3}$, and the area of $\triangle s_1 v_3 v_4$ can be computed as $\frac{1}{4} d(s_1, s_4) d(v_3, v_4)$. The area of the Voronoi cell for a node is the sum of the area of such Voronoi triangles contained within the Voronoi cell. However, the exact area of the uncovered portion of a Voronoi cell is not equal to the area of this Voronoi cell minus the area of the sensing disk. This is because the sensing disk of a sensor node may protrude its Voronoi cell. For example, in Fig. 8.1, some of the s_1's sensing disk are also located at the s_4's Voronoi cell. The protrusion depends on the relations between the sensing range and the distance between two Voronoi neighbors and the lengths of the Voronoi triangle sides. The calculation for

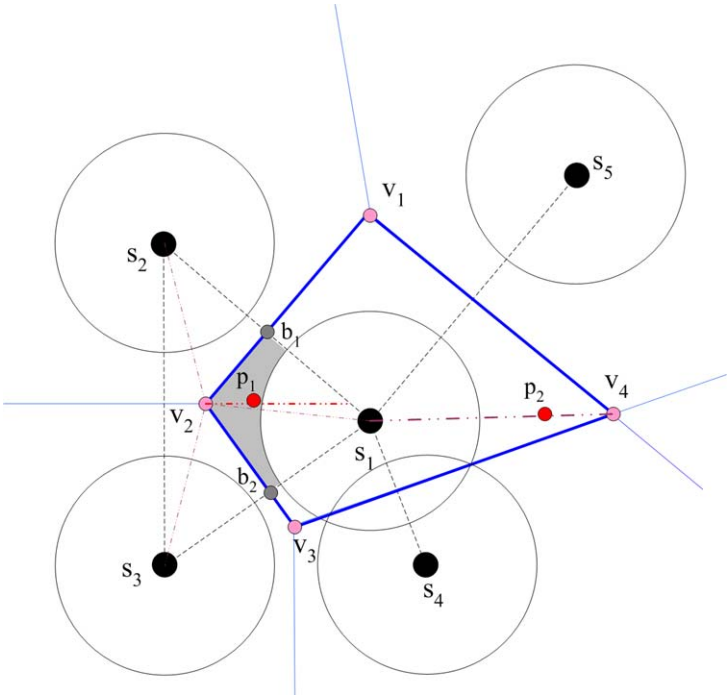

Fig. 8.2 Illustration of determining mobile nodes' destinations

the exact area of uncovered portion within a Voronoi cell is complicated. Wang et al. [23] propose to use the distance between itself and its furthest Voronoi vertex to decide a coverage hole and the size of the coverage hole. If such a distance is larger than its sensing range, then there exists a coverage hole, and the size of the hole is considered as proportional to the distance.

Destination Selection After deciding the existence of a coverage hole and its size, a stationary node needs to decide the number of mobile nodes and the target locations of these mobile nodes to heal its holes. Ghosh [7] proposes that for each Voronoi vertex, one mobile node should be used to heal the coverage hole around this Voronoi vertex if the size of its coverage hole within the Voronoi cell is larger than a threshold. For example, in Fig. 8.2, the shaded area is the uncovered area around the Voronoi vertex v_2 within the node s_1's Voronoi cell. If its size is larger than a threshold $\rho\pi R_s^2$ $(0 < \rho \leq 1)$, then one mobile node is needed to heal the coverage hole around v_2. Furthermore, Ghosh [7] proposes the following conditions to determine the destination of a mobile node. The target location p_1 (1) lies on the line that bisects the angle formed by v_2 and the adjacent Voronoi edges $\overline{v_2b_1}$ and $\overline{v_2b_2}$, (2) lies within the node v_2's Voronoi cell, and (3) $d(s_1, p_1) = \min\{2R_s, d(s_1, v_2)\}$. Wang et al. [23] propose that only one mobile node is used to heal the coverage hole of a Voronoi cell. The target location p_2 lies on the line connecting the sensor node and its furthest Voronoi vertex, and the distance $d(s_1, p_2) = \max\{\sqrt{3}R_s, d(s_1, v_4)\}$.

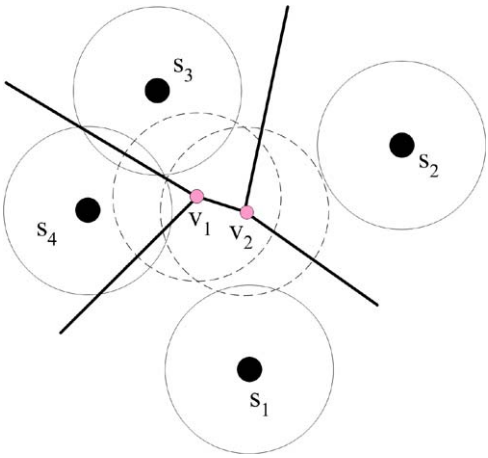

Fig. 8.3 Illustration of duplicate healing

Movement Strategy After the selection of target locations, a stationary node can then *bid* mobile nodes to heal its coverage holes by broadcasting its demand of mobile nodes and their target locations [7, 23]. In order to reduce message overhead, a stationary node does not flood its demand message to the whole network but broadcasts its demand only to its 1-hop and 2-hop neighbors instead. Due to the limited demand advertisement radius, a stationary node only has incomplete information about the mobile nodes and may cause a *duplicate healing* problem: Different mobile nodes accept their bids from different stationary nodes but move to close locations to heal the same coverage hole. This often happens when two target locations are very close and less than the sensing range. Figure 8.3 illustrates an example of duplicate healing. The node s_1 requires a mobile node to move to its furthest Voronoi vertex v_1, while the node s_2 requires a mobile node to move to its furthest Voronoi vertex v_2. If two mobile nodes move to v_1 and v_2, respectively, then a duplicate healing occurs. To avoid duplicate healing, one may put a distance constraint on two target locations. If the distance is less than a threshold, then only one of the locations will be selected as the mobile destination.

A mobile node that has received more than one bid selects the best bid and moves to the destination specified by this bid. A bid contains the information of the hole size and target location. Hence before moving, a mobile node can calculate its moving cost such as the traverse distance and its moving benefit such as the hole size and select the best bid that has the highest overall price (benefit minus cost). After mobile nodes have made their first movements, some coverage holes that are caused solely by stationary nodes will be repaired. However, after one round of movement, there may still have some coverage holes which are caused by both stationary and mobile nodes. In such a case, a second round for hole detection and advertisement and mobile node movement is needed. In this second and afterward round, since mobile nodes already have some contributions in coverage, they need to include the leaving cost into the price calculation. The leaving cost is the size of the hole that it will cause if it moves out of its current location. By this iterative movements, mobile nodes will move to heal larger and larger holes. However, mobile nodes are

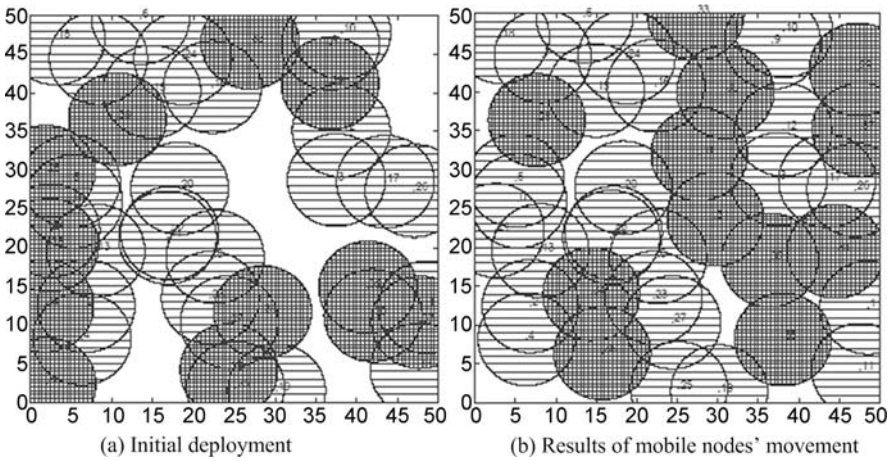

Fig. 8.4 An example of using mobile nodes to heal coverage holes (reproduced from [19], ©2004, IEEE)

likely to move in an irregular pattern in such iterative movements and consumes more energy than moving directly from their initial locations to final destinations. This problem can be solved by using virtual movement or logical movement. Instead of moving physically in each round, mobile nodes perform virtual movements once they choose a bid. After several rounds of bidding, mobile nodes can determine their final destinations and only make their physical movements to their final destinations. This approach reduces the overall movement distance with the cost of the increased message overheads. The proxy-based bidding protocol proposed by Wang et al. [23] describes an implementation of such virtual movement strategy. We close this section with an example taken from Wang et al. [19] to illustrate how coverage holes can be repaired by mobile nodes. Figure 8.4 plots the initial deployment of 28 stationary nodes and 12 mobiles nodes. The stripped shadow is the sensing area of stationary sensors, and the gridded shadow is that of mobile sensors. Figure 8.4 shows how mobile nodes heal the coverage holes.

8.2 Optimizing Area Coverage

In one random network deployment, it is not uncommon that the covered area of a sensor overlaps others, even if nodes are uniformly scattered. In a mobile sensor network where all nodes can move around, the mobile nodes can adjust their positions after initial deployment in order to reduce their coverage overlaps and maximize area coverage. The coordination of mobile nodes' movement is a much challenging issue in a mobile sensor network. A node's movement may change the already covered area of itself and its neighbors and become a cause for other nodes to move, which may cause the oscillations of nodes' movement and nodes may never

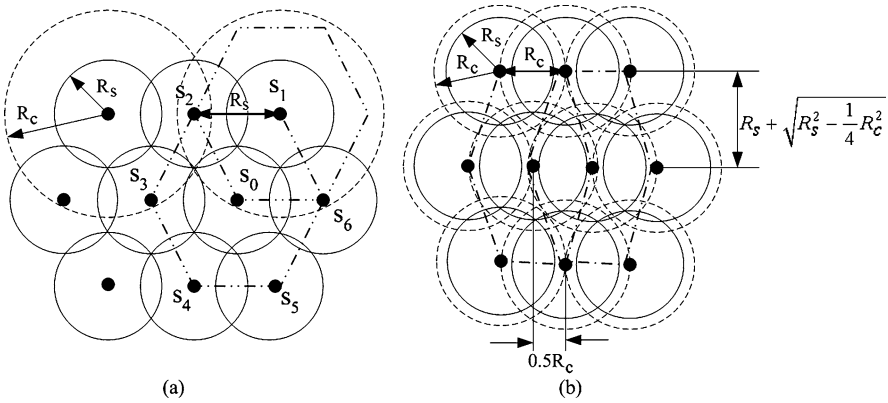

Fig. 8.5 Illustration of using coverage pattern to determine mobile target locations

be stopped. This is in contrast to the hole heal problem where mobile nodes are to move to the coverage holes caused by the stationary nodes.

In a mobile sensor network, one of the movement objectives is to maximize area coverage. That is, after the nodes' movement, their covered area can be greatly increased compared to that of before nodes' movement. Some other objectives include optimizing movement cost and reducing message overhead. When designing a node movement algorithm for a mobile network, the following issues need to be addressed. First, where to relocate a mobile node and when should a mobile node be stabilized at a location. Second, how to dispatch mobile nodes to the target locations while minimizing the moving and messaging cost. This section groups the existing movement schemes into three categories, namely, coverage pattern based movement, virtual force based movement, and grid quorum based movement, and shows how these schemes approach the design issues.

8.2.1 Coverage Pattern Based Movement

In the group of coverage pattern-based node movement strategies [3, 18, 22, 25], the target locations for mobile nodes are computed based on a predefined coverage pattern that can meet both area coverage and network connectivity requirements. A coverage pattern can be either global or local. A global coverage pattern actually is a deterministic sensor placement plan which specifies all target locations in a sensor field. A local coverage pattern of a selected node specifies only the target locations surrounding itself for other mobile nodes. In both cases, a coverage pattern can be regarded as to use polygons to tile up the whole sensor field, and the vertices of these polygons are the target locations for mobile nodes.

The most commonly used coverage pattern is the regular hexagon with the sensing range R_s as its side length. As illustrated in Fig. 8.5(a), the sensor field can

be completely covered by tiling-up such regular hexagons, and hence the target locations for mobile nodes are the vertices of these hexagons. Furthermore, if the transmission range $R_c \geq \sqrt{3}R_s$, then the nodes located at hexagons' vertices can form a connected network. In [22], nodes' movement is based on a local hexagon coverage pattern. At first, one node is selected as a seed. It computes six hexagon locations to form a regular hexagon surrounding itself and with side length R_s. For each hexagon vertex, the seed node greedily selects, among its neighbors, the one with the shortest distance to this hexagon vertex as its target location. A node which has just moved to its target location becomes a seed node and performs a new round of target location computation and mobile node selection.

Wang et al. [25] propose the following coverage pattern when $R_c < \sqrt{3}R_s$, as illustrated in Fig. 8.5(b). Sensors on each row are separated by a distance of R_c so that the connectivity of each row is guaranteed. Adjacent rows are separated by a distance of $R_s + \sqrt{R_s^2 - \frac{1}{4}R_c}$ and shifted by a distance of $\frac{1}{2}R_c$. The sensor field can be completely covered by such a coverage pattern. To ensure network connectivity, an additional column of sensors need to be added, and the sensors in the column are separated by a distance not larger than R_c. After computing the target locations, Wang et al. [25] convert the sensor movement problem into a maximum-weight maximum-matching problem to decide which mobile node should move to which target location. Let $\mathcal{S} = \{s_1, \ldots, s_n\}$ denote the set of sensors' locations, and $\mathcal{L} = \{l_1, \ldots, l_m\}$ the set of target locations computed by the coverage pattern. It is assumed that the number of mobile nodes is larger than the number of target locations, i.e., $|\mathcal{S}| > |\mathcal{L}|$. The solution involves the following steps:

1. Construct a weighted complete bipartite graph $\mathcal{G} = (\mathcal{S} \cup \mathcal{L}, \mathcal{S} \times \mathcal{L})$. If the objective is to minimize the total energy consumption, the weight of an edge (s_i, l_j) can be defined as

$$w(s_i, l_j) = -\Delta_m \times d(s_i, l_j).$$

   If the objective is to maximize the average remaining energy of sensors, the weight can be defined as

$$w(s_i, l_j) = e_i - \big(\Delta_m \times d(s_i, l_j)\big).$$

   In the above, e_i is the current energy of a sensor s_i, Δ_m is the energy consumption to move a sensor a unit distance, and $d(s_i, l_j)$ is the distance between a sensor and a target location.
2. (Optional) Extend the graph $\mathcal{G}$ to $\mathcal{G}'$ if $|\mathcal{S}| > |\mathcal{L}|$. The new graph $\mathcal{G}' = (\mathcal{S} \cup \mathcal{L} \cup \mathcal{L}', S \times \{\mathcal{L} \cup \mathcal{L}'\})$ consists of a set of virtual locations $\mathcal{L}'$ ($|\mathcal{L}'| = |\mathcal{S}| - |\mathcal{L}|$). The virtual locations are used to make equal size of the two sides of the bipartite graph. New edges are then added from each s_i to each $l_j \in \mathcal{L}'$ with weight

$$w_{\min} = \min_{s_i \in \mathcal{S}, l_k \in \mathcal{L}} \big\{w(s_i, l_k)\big\} - 1.$$

3. Solve the maximum-weight perfect-matching problem on $\mathcal{G}'$. The purpose is to find a perfect matching $\mathcal{M}$ in $\mathcal{G}'$ with the maximum edge weights. The Hungarian method [13] can be used to find $\mathcal{M}$.

4. Move nodes to their respective target locations. If an edge (s_i, l_j) exists in $\mathcal{M}$ and $l_j \in \mathcal{L}$, then move node to l_j.

In [25], Wang et al. also consider how to move sensors in a sensor field with arbitrarily shaped obstacles, and in [18], Wang and Tseng extend their coverage pattern-based movement strategy to dispatch mobile nodes for k-coverage.

8.2.2 Virtual Force Based Movement

In the group of virtual force-based node movement strategies [1, 10, 11, 15, 21, 29], mobile nodes are likened as electromagnetic particles. Similar to the electrostatic force between two electromagnetic particles in the physical world, two mobile nodes as two particles expel each other if their distance is too close or attract each other if their distance is far apart. A mobile node may also experience other types of force such as attractive force of the sensor field boundaries and the repulsive forces by obstacles. A force can be described as a vector, and the overall force is then the result of vector addition. The overall force specifies the direction and distance that a mobile node should move to. Generally speaking, a type of force can be expressed as a function of the distance between the node in consideration and other nodes. Therefore, the calculation of the direction and magnitude of a force is based on the assumption of known locations of a node and its neighbors. Sometimes, other constraints such as the density of neighborhood, the maximal enforcing distance, the number of 1-hop neighbors, etc. are also included in a force function.

For example, Heo and Varshney [10] define the following conditions for a force function $f(\cdot)$:

1. Inverse relation: $f(d_{ij}) \geq f(d_{ik})$ if $d_{ij} \leq d_{ik}$. d_{ij} and d_{ik} are the Euclidean distances from sensor s_i to sensors s_j and s_k, respectively. This implies that the force is inversely proportional to nodes' distance.
2. Upper bound: $f(0^+) = f_{\max}$, where $f_{\max}$ is the maximum force. This sets an upper bound on forces.
3. Lower bound: $f(d_{ij}) = 0$ if $d_{ij} > R_c$, where R_c is the communication range. This indicates that only neighbors exert forces.

Sensor nodes are moved step by step in [10]. In step n, the force on sensor s_i from another sensor s_j within its communication range ($d_{ij} < R_c$) is calculated to be the repulsive force

$$f_{ij}^{(n)} = \frac{D_i^{(n)}}{\mu^2} \times \left(R_c - d_{ij}^{(n)}\right) \times \frac{p_j^{(n)} - p_i^{(n)}}{d_{ij}^{(n)}},$$

where $p_i^{(n)}$ is the location of sensor s_i at step n, $D_i^{(n)}$ the local density of s_i at step n, and μ is the expected density of an ideal deployment. The inclusion of the density factor $\frac{D_i^{(n)}}{\mu^2}$ with larger repulsive for a node with higher local density is expected to

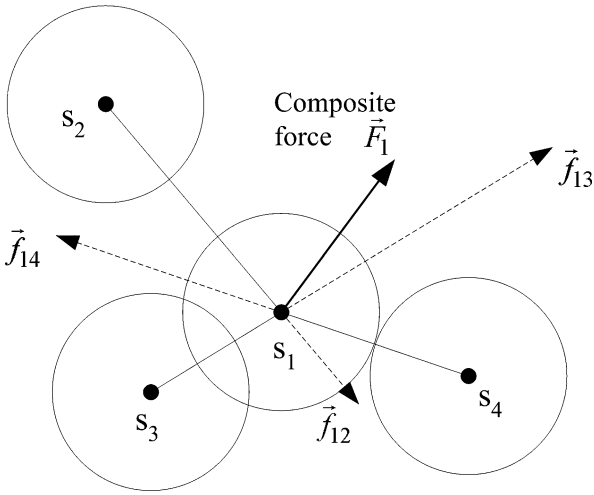

Fig. 8.6 Illustration of using virtual repulsive forces to determine the target location

expedite the process of node spreading. Figure 8.6 presents an example of how to use the composite force to determine the target location for a mobile node. In the figure, three sensors exert repulsive forces to sensor s_1, and the composite force is represented by $\overrightarrow{F_1}$, whose end point specifies the target location for sensor s_1. Heo and Varshney [10] also introduce two rules to stop a node's movement.

1. Oscillation check: If a node moves back and forth between similar locations many times, then this node is regarded to be in the oscillation state. If the number of oscillations exceed a predefined threshold, then this node should stop and locate itself at the gravity center of its oscillation points.
2. Stability check: If the moving distance of a node is less than a predefined threshold for several times, then this node can be considered to have reached a stable status, and it should stop its further movement.

Zou and Chakrabarty [29] propose that a sensor s_i can experience three types of forces: The overall attractive (positive) force $\overrightarrow{F_{iA}}$ due to preferential coverage areas, the overall repulsive (negative) force $\overrightarrow{F_{iR}}$ by all obstacles, and the forces $\overrightarrow{f_{ij}}$ exerted by another sensor s_j. Therefore, the total force $\overrightarrow{F_i}$ can be expressed as

$$\overrightarrow{F_i} = \overrightarrow{F_{iA}} + \overrightarrow{F_{iR}} + \sum_{j \neq i} \overrightarrow{f_{ij}}.$$

The force $\overrightarrow{f_{ij}}$ is expressed by the polar coordinate notation (r, θ), where r is the magnitude, and θ is the orientation angle.

$$\overrightarrow{f_{ij}} = \begin{cases} (w_a(d_{ij} - d_{\text{th}}), \alpha_{ij}) & \text{if } d_{ij} > d_{\text{th}}, \\ 0 & \text{if } d_{ij} = d_{\text{th}}, \\ (\frac{w_r}{d_{ij}}, \pi + \alpha_{ij}) & \text{if } d_{ij} < d_{\text{th}}, \end{cases}$$

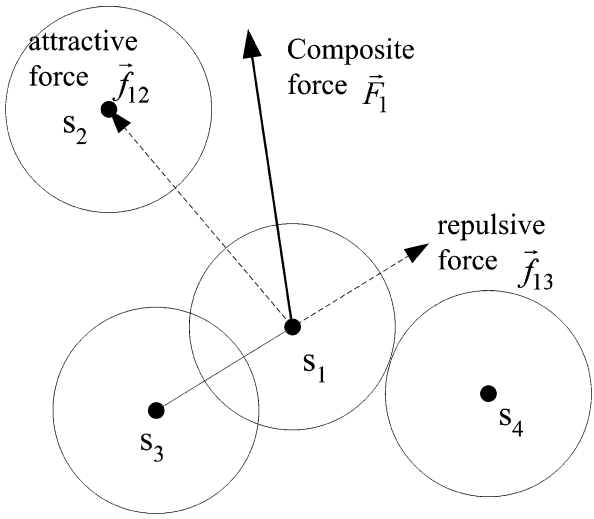

Fig. 8.7 Illustration of using virtual attractive and repulsive forces to determine the target location

where d_{ij} is the Euclidean distance between s_i and s_j, d_{th} is a distance threshold, α_{ij} is the orientation angle of a line segment from s_i to s_j, and w_a and w_r are measures of the attractive and repulsive forces, respectively. The distance threshold d_{th} controls how close sensors get to each other. Figure 8.7 illustrates an example. The distance threshold is set to the distance d_{14}, and hence sensor s_4 does not exert any force to sensor s_1. Since $d_{12} > d_{\text{th}}$, sensor s_2 exerts an attractive force $\overrightarrow{f_{12}}$, and since $d_{13} < d_{\text{th}}$, sensor s_3 puts a repulsive force $\overrightarrow{f_{13}}$. The composite force is $\overrightarrow{F_1}$, and its end point specifies the target location for sensor s_1. Zou and Chakrabarty [29] define the maximum times that can be used for nodes' movement. All nodes should stop their movements if they have moved $n_{\max}$ times. Furthermore, if the area coverage requirement has already been met before $n_{\max}$ movements, all nodes should also stop their movements. To save moving cost, Zou and Chakrabarty propose to use virtual movement: In each step, all nodes only exchange their target locations other than make physical movements. The physical movement is only done once for each node and after all nodes have decided their final locations. The virtual movement reduces moving cost but is with the cost of increased message overhead.

8.2.3 Grid Quorum Based Movement

In the group of grid quorum-based node movement strategies [4, 5, 12, 20, 26–28], the mobility-assisted network redeployment problem is viewed as a load balancing problem in traditional parallel processing systems. The sensor field is partitioned into many small grid cells, and the number of nodes in each cell is considered as the load of the cell. Unlike the coverage pattern-based and the virtual force-based movement schemes, the grid quorum-based movement schemes need not to specify

the exact target location for a moving node. Instead, a target cell is specified for a moving node, and this node can move to any location within the target cell.

The size of a grid cell should be determined according to the coverage and connectivity requirement. To ensure complete coverage of a grid cell by any sensor within a grid cell, the side length of the grid cell should be $l \leq R_s/\sqrt{2}$. To ensure that any node within a cell can directly communicate with any other node in its four adjacent cells, the transmission range of a node should be at least the diagonal of the rectangle constructed from two adjacent cells, and hence $l \leq R_c/\sqrt{5}$. Therefore, the cell size can be chosen as $l = \min\{\frac{R_s}{\sqrt{2}}, \frac{R_c}{\sqrt{5}}\}$.

The objective of node movement is to achieve a load balanced state where the difference between the maximum cell load (the number of nodes in a cell) and the minimum cell load is at most 1. To achieve a balanced state, mobile nodes in overloaded cells should move to underloaded cells. Furthermore, nodes should move in an efficient way so that the total moving distance, the total moving energy consumption, and/or the moving delay can be minimized. It is also desirable that such moving costs can be shared among nodes in a fair way. A movement plan states, for each cell, the number of sensors to be moved out of the cell, and the target cell for each moving-out node. An optimal movement plan minimizing the total moving distance can be obtained, however, normally with a centralized approach.

Wu and Yang [26] propose to convert the grid load balancing problem into a perfect matching problem over a bipartite graph and use a Hungarian method to obtain the optimal movement plan. In the bipartite graph, the left-side nodes are those overloaded cells, and the right-side nodes are those underloaded cells; and the edges are the moving distance from one cell to another. The optimal solution is generally assumed to be computed by the sink, which needs to first collect global information and then disseminates the movement plan to all nodes. However, a centralized approach cannot adapt to the dynamics of the network (some nodes may suddenly die) and is not easy to implement in large-scale WSNs. In the literature, many localized algorithms have been proposed.

Yang et al. [28] apply a scan technique (called scan-based movement-assisted sensor deployment method, SMART) to identify overloaded and underloaded cells and to direct the movement of those moving nodes from overloaded cells. The scan process will be executed twice–once for all rows and once for all columns—to achieve a balanced state. The scan procedure for a row is as follows. It is first initiated from one end of the row to another and then from the other end back to the initial end (second scan). After the first scan, the average number of nodes for each cell in a balanced state can be obtained. After the second scan, each cell can identify whether it is overloaded or underloaded or neutral. Let w_i denote the load of a cell i, and let v_i denote the prefix sum of the first i cells, i.e., $v_i = \sum_{j=1}^{i} w_j$. After the first scan from left to right, the right end of the row computes the average load of a cell as $\overline{w} = v_n/n$. In the second scan from right to left, each cell compares its load w_i with $\overline{w}$. Specifically, if $w_i = \overline{w}$, then cell i is in a *neutral* state; if $w_i > \overline{w}$, then cell i is overloaded and in a *give* state; if $w_i < \overline{w}$, then cell i is underloaded and in a *take* state. A node in a give state cell needs also to determine the number of sensors to be sent to each direction. Let $w_i^{\rightarrow}$ denote the number of sensors to be sent from a give

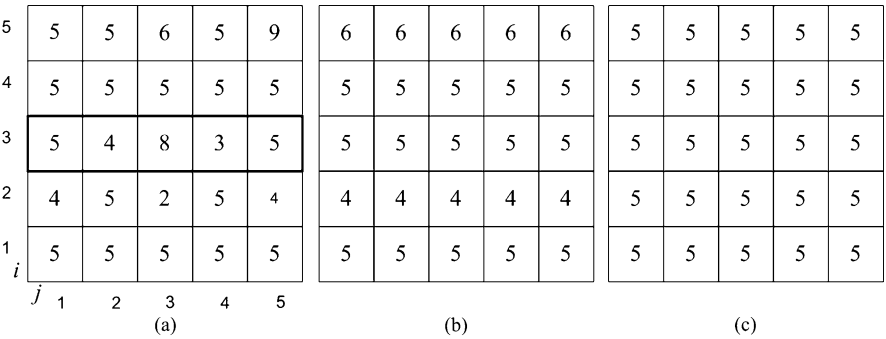

Fig. 8.8 Illustration of a SMART example: (**a**) Initial node deployment; (**b**) after row scan; and (**c**) after column scan

state cell i to the right direction, and ${}^{\leftarrow}w_i$ to the left direction. They are computed as

$$w_i^{\rightarrow} = \min\{w_i - \overline{w}, \max\{v_i - \overline{v}_i, 0\}\},$$
$${}^{\leftarrow}w_i = (w_i - \overline{w}) - w_i^{\rightarrow},$$

where $\overline{v}_i = i\overline{w}$ is the prefix sum in the balanced state.

After determining $w_i^{\rightarrow}$ and ${}^{\leftarrow}w_i$, a simple sender-initiated node dispatch algorithm [28] then is executed to dispatch mobile nodes from overloaded cells to underloaded cells as follows:

1. For each cell i in the give state, the cell head sends $w_i^{\rightarrow}$ sensors to its right neighbor and ${}^{\leftarrow}w_i$ sensors to its left neighbor.
2. For each cell j in the take state, when the cell head senses several bypassing sensors, it intercepts as many sensors as possible to fill its deficiency, $\overline{w} - w_j$, and let the superfluous sensors move along their respective original direction.

This approach is simple in that a cell in the take state needs not to distinguish how many sensors should be taken from its right side and how many taken from its left side. The scan algorithm can be illustrated by an example taken from [28]. Figure 8.8(a) is the initial unbalanced state. After the first row scan from left to right, the average load of the third row is $\overline{w} = 5$, and only the cell $(3, 3)$ is in the give state. Furthermore, after the second row scan from right to left, it computes $w_3^{\rightarrow} = 2$ and ${}^{\leftarrow}w_3 = 1$. After applying the sender-initiated node dispatch algorithm, all rows achieve the balanced state, as shown in Fig. 8.8(b). The same procedure is then applied to all columns to achieve balanced columns, as shown in Fig. 8.8(c).

In the design of a node movement strategy, it is often assumed that given enough remaining energy, a mobile node can move to a desired location without any limitation in the movement distance. In some cases, however, mobile nodes can only move within a limited distance. In [4, 5], Chellappan et al. argue that a kind of mobile nodes can only *flip* a limited distance to a new location and can only flip once. For example, the locomotion unit of a node can be coiled springs, and a node

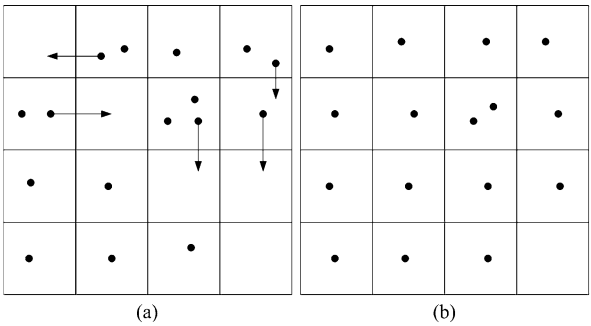

Fig. 8.9 Illustration of flip plan A: (**a**) Initial node deployment and flip plan; (**b**) the resulting deployment, the right bottom cell not covered. The total number of flips is 5

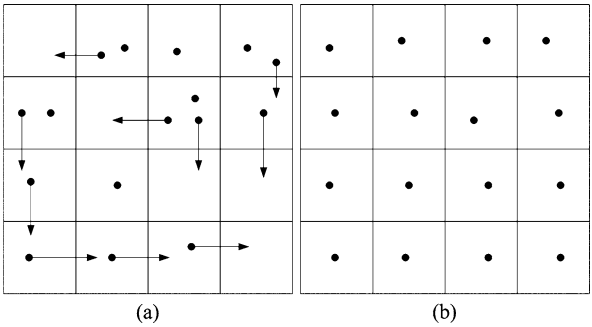

Fig. 8.10 Illustration of flip plan B: (**a**) Initial node deployment and flip plan; (**b**) the resulting deployment, all cells covered. The total number of flips is 10

moves/flips when it unwind its springs. The initial deployment of such flip-based sensor nodes may not provide optimal coverage of the sensor field. Therefore, the objective is to design an optimal flip plan such that after nodes flip, the area coverage can be maximized, and the total number of flips is minimized. However, finding such an optimal flip plan is not trivial. We use the example taken from [5] to illustrate this. As shown in Fig. 8.9, the flip plan A cannot cover all cells, however, it only requires five flips. The flip plan B as shown in Fig. 8.10 can cover all cells, but it requires more flips.

Chellappan et al. translate the problem of finding an optimal flip plan into a minimum-cost maximum-flow problem. A cell is called a *source* if it has two or more nodes; a cell is a *forwarder* if it has only 1 node; and a cell is a *sink* if it has no node. The maximum flow problem is to maximize flows from multiple sources to multiple sinks in a graph without violating capacity constraint of any graph edge. Each flow in the graph denotes a flip path from a source to a sink, and the maximum flow value denotes the maximum number of empty cells that can be filled. To minimize the number of total flips, each flip is also associated with a cost. Then the minimum-cost maximum-flow problem is to find paths that minimize the overall cost while still maximizing the flows. Many existing algorithms can be used to solve a minimum-cost maximum-flow problem in a graph.

Chellappan et al. propose to construct a virtual graph as follows. Each cell is represented by three vertices: The *base* vertex (v^b) tracks the number of nodes in a cell, the *in* vertex (v^i) tracks the numbers of nodes that have flipped into this cell,

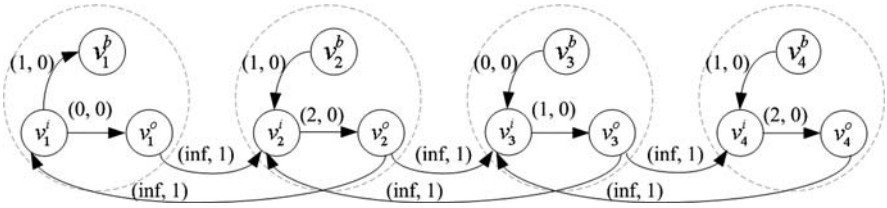

Fig. 8.11 An illustration of a virtual graph for the four cells in *the top row* of Fig. 8.9. The (capacity, cost) is shown for each edge

and the *out* vertex (v^o) tracks the number of nodes that have flipped out of this cell. Edges are then added in order to encourage nodes to flip from source cells to sink cells. Suppose that there are n nodes in a cell at the initial network deployment. For the three vertices in one cell, an edge $\langle v^i, v^o \rangle$ is added with capacity n from vertex v^i to v^o, and an edge $\langle v^o, v^i \rangle$ with infinite capacity *inf* is added. If $n > 0$, an edge $\langle v^b, v^i \rangle$ with capacity $n - 1$ is added, and if $n = 0$, an edge $\langle v^i, v^b \rangle$ with capacity 1 is added. For vertices in different cells, if a node can flip from a cell i to cell j, then an edge $\langle v_i^o, v_j^i \rangle$ with infinite capacity *inf* is added. The cost of an edge needs to represent the number of flips. Therefore, a cost of 1 is added for the edges connecting two cells (e.g., $\langle v_i^o, v_j^i \rangle$), and a cost of 0 is added for all other edges. Figure 8.11 illustrates a virtual graph constructed for the four cells in the top row in Fig. 8.9. After constructing the virtual graph, the first step is to determine the maximum flow value from all base vertices of source cells to base vertices of sink cells in the graph. Then the second step is to determine the minimum-cost flow plan achieving such maximum flow value. Many existing algorithms can be used to find minimum-cost maximum-flow paths. For example, the Edmonds–Karp algorithm [6] can be used to find the maximum flow value, and the successive approximation algorithm [8, 9] can be used to find minimum-cost flow plan. Each flow path specifies a flip path. For example, a flow path $\langle v_i^b, v_i^i, v_i^o, v_k^i, v_k^o, v_l^i, v_l^o, \ldots, v_m^o, v_j^i, v_j^b \rangle$ denotes the flip sequence $\langle i, k, l, \ldots, m, j \rangle$ where one node should flip from cell i to cell k, one node from k to i, etc.

8.3 Improving Event Coverage

In an event detection and monitoring sensor network, sensing tasks are performed in an on-demand manner. At first, sensors are used to detect events which may appear and disappear at certain places within a sensor field. After detecting an event, it is often desirable to relocate more mobile sensors close to the event location for performing the sensing task. In our context, the location of an event will not change during its lifetime. This is different from the object tracking problem where the event source (the object) moves in the sensor field. The event coverage problem can be simply expressed as how to dispatch which mobile nodes to which event sources.

Wang et al. [24] consider a hybrid network consisting of both stationary and mobile sensors. It is assumed that all stationary sensors provide complete area coverage

and can detect events. After detecting an event, one mobile node needs to be sent to the event source for more advanced sensing and analysis tasks. In [24], the event coverage problem is first modeled as a maximum-matching problem in a weighted bipartite graph. The objective is to find an optimal match from the set of the mobile nodes' locations to the set of the events' locations such that not only the total moving energy but also the standard deviation of energy consumption among mobile nodes can be minimized. This is done by introducing a preference list and a bound to avoid extreme cases. Furthermore, when the number of mobile nodes n is less than the number of the event sources m, they propose to first group these m event locations to only n clusters in constructing a weighted bipartite graph and then let a mobile node travel all the event locations within the assigned cluster. This algorithm is centralized and needs a lot of computation. Wang et al. [24] also propose a grid-quorum-based method similar to that in [20] to dispatch mobile nodes to event cells (those grid cells contain events).

Butler and Rus [2] consider the event coverage problem in a mobile sensor network. The objective is to dispatch more mobile sensors to event locations while still maintaining complete coverage of the sensor field. In order to approximate event distributions by mobile nodes, two moving strategies are proposed, namely, the *history-free strategy* and the *history-based strategy*. In the history-free strategy, each sensor reacts to an event by moving according to a function of the form

$$s_i^{k+1} = s_i^k + f\left(l^{k+1}, s_i^k, s_i^0\right),$$

where l^{k+1} is the position of event $k+1$, s_i^0 is the sensor initial location, and s_i^k and s_i^{k+1} are the positions of sensor i before and after event $k+1$. The function $f(\cdot)$ depends on the distance d between a sensor and an event, and the following three criteria are proposed for f:

1. $0 \le f(d) \le d$ for all d. This indicates that after an event occurs, a sensor should never move past that event.
2. $f(\infty) = 0$. This suggests that the sensors' movement should tend to 0 as the event gets further away.
3. $f(d_1) - f(d_2) < d_1 - d_2$ for all $d_1 > d_2$. This means that no sensor should move past another along the same vector in response to the same event.

An example function f is $f(d) = \alpha d^\beta e^{-\gamma d}$ for values of parameters α, β, γ such that $\alpha e^{-\gamma d}(\beta d^{\beta-1} - \gamma d^\beta) > 1$ for all d. In the history-based strategy, each sensor needs to maintain event history to improve the sensor's approximation of the event distribution. Each sensor maintains a discrete version of the *cumulative distribution function* (CDF) of the events, which is updated after each event. Let l_x^k denote the x-coordinate of the event k. Each sensor updates its event CDF as follows:

1. Increment CDF bins representing positions $\le l_x^k$.
2. Scale CDF by $k/(k+1)$.
3. Find bins b_i and b_{i+1} with value $b_i \le x_0 \le b_{i+1}$, where x_0 is the initial x-coordinate of the sensor.

4. Compute the final x-coordinate position of the sensor by interpolating the values of b_i and b_{i+1}.

The above steps are also applied to the y-coordinate of each sensor.

The above two strategies move sensors closer to event locations. In order to maintain complete coverage of the sensor field, Butler and Rus [2] propose to use a Voronoi diagram to determine whether the movement of a sensor will cause a coverage hole. If any part of the sensor's Voronoi cell is farther away than the sensing range, the sensor knows that its moving will cause a coverage hole,. and in this case, the sensor should stop moving to maintain its original coverage. To save control message overhead, a sensor can construct its Voronoi cell only based on its 1-hop neighbors.

References

1. Akkaya, K., Janapala, S.: Maximizing connected coverage via controlled actor relocation in wireless sensor and actor networks. Computer Networks (Elsevier) **52**(14), 2779–2796 (2008)
2. Butler, Z., Rus, D.: Event-based motion control for mobile-sensor networks. IEEE Pervasive Computing Magazine **2**(4), 34–42 (2003)
3. Chang, C.Y., Chang, H.R., Hsieh, C.C., Chang, C.T.: Ofrd: Obstacle-free robot deployment algorithms for wireless sensor networks. In: IEEE Wireless Communications and Networking Conference (WCNC), pp. 4371–4376 (2007)
4. Chellappan, S., Gu, W., Bai, X., Xuan, D., Ma, B., Zhang, K.: Deploying wireless sensor networks under limited mobility constraints. IEEE Transactions on Mobile Computing **6**(10), 1142–1157 (2007)
5. Chellappan, S., Bai, X., Ma, B., Xuan, D., Xu, C.: Mobility limited flip-based sensor networks deployment. IEEE Transactions on Parallel and Distributed Systems **18**(2), 199–211 (2007)
6. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. 2nd Edition. MIT Press, Cambridge (2001)
7. Ghosh, A.: Estimating coverage holes and enhancing coverage in mixed sensor networks. In: IEEE International Conference on Local Computer Networks, pp. 68–76 (2004)
8. Goldberg, A.V.: An efficient implementation of a scaling minimum-cost flow algorithm. Journal of Algorithms **22**(1), 1–29 (1997)
9. Goldberg, A.V., Tarjan, R.E.: Solving minimum-cost flow problems by successive approximation. In: ACM Annual Symposium on Theory of Computing (STOC), pp. 7–18 (1987)
10. Heo, N., Varshney, P.K.: Energy-efficient deployment of intelligent mobile sensor networks. IEEE Transactions on Systems, Man and Cybernetics, Part A **35**(1), 78–92 (2005)
11. Howard, A., Matarić, M.J., Sukhatme, G.S.: Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In: The 6th International Symposium on Distributed Autonomous Robotics Systems (DARS), pp. 1–10 (2002)
12. Jiang, Z., Wu, J., Kline, R., Krantz, J.: Mobility control for complete coverage in wireless sensor networks. In: The 28th International Conference on Distributed Computing Systems Workshops (ICDCS), pp. 291–296 (2008)
13. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistic Quarterly **2**, 83–97 (1955)
14. Manoj, B., Sekhar, A., Murthy, C.S.R.: On the use of limited autonomous mobility for dynamic coverage maintenances. Computer Networks (Elsevier) **51**(8), 2126–2143 (2007)
15. Poduri, S., Sukhatme, G.S.: Constrained coverage for mobile sensor networks. In: IEEE International Conference on Robotics and Automation, pp. 165–172 (2004)

16. Sibley, G.T., Rahimi, M.H., Sukhatme, G.S.: Robomote: A tiny mobile robot platform for large-scale ad-hoc sensor networks. In: IEEE International Conference on Robotics and Automation, pp. 1143–1148 (2002)
17. Tseng, Y.C., Wang, Y.C., Cheng, K.Y., Hsieh, Y.Y.: iMouse: An integrated mobile surveillance and wireless sensor system. IEEE Computer **40**(6), 76–82 (2007)
18. Wang, Y.C., Tseng, Y.C.: Distributed deployment schemes for mobile wireless sensor networks to ensure multi-level coverage. IEEE Transactions on Parallel and Distributed Systems (2008)
19. Wang, G., Cao, G., LaPorta, T.: Proxy-based sensor deployment for mobile sensor networks. In: IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), pp. 493–502 (2004)
20. Wang, G., Cao, G., Porta, T.L., Zhang, W.: Sensor relocation in mobile sensor networks. In: IEEE Infocom, pp. 2302–2312 (2005)
21. Wang, G., Cao, G., Porta, T.F.L.: Movement-assisted sensor deployment. IEEE Transactions on Mobile Computing (ToMC) **5**(6), 640–652 (2006)
22. Wang, P.C., Hou, T.W., Yan, R.H.: Maintaining coverage by progressive crystal-lattice permutation in mobile wireless sensor networks. In: IEEE International Conference on Systems and Networks Communication (ICSNC), pp. 1–6 (2006)
23. Wang, G., Cao, G., Berman, P., Porta, T.F.L.: Bidding protocols for deploying mobile sensors. IEEE Transactions on Mobile Computing **6**(5), 515–528 (2007)
24. Wang, Y.C., Peng, W.C., Chang, M.H., Tseng, Y.C.: Exploring load-balance to dispatch mobile sensors in wireless sensor networks. In: IEEE the 16th International Conference on Computer Communications and Networks (ICCCN), pp. 669–674 (2007)
25. Wang, Y.C., Hu, C.C., Tseng, Y.C.: Efficient placement and dispatch of sensors in a wireless sensor network. IEEE Transactions on Mobile Computing **7**(2), 262–274 (2008)
26. Wu, J., Yang, S.: Optimal movement-assisted sensor deployment and its extensions in wireless sensor networks. Simulation Modelling Practice and Theory (Elsevier) **15**(4), 383–399 (2007)
27. Yang, S., Wu, J., Dai, F.: Localized movement-assisted sensor deployment in wireless sensor networks. In: IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS), pp. 753–758 (2006)
28. Yang, S., Li, M., Wu, J.: Scan-based movement-assisted sensor deployment methods in wireless sensor networks. IEEE Transactions on Parallel and Distributed Systems **18**(8), 1108–1121 (2007)
29. Zou, Y., Chakrabarty, K.: Sensor deployment and target localization in distributed sensor networks. ACM Transactions on Embedded Computing Systems **3**(1), 61–91 (2004)

Part IV
Barrier Coverage Problems

Chapter 9
Build Intrusion Barriers

Abstract Intrusion detection is a typical application in wireless sensor networks. When mobile objects are entering into the boundary of a sensor field or are moving cross the sensor field, they should be detected by the deployed sensors. A mobile object may enter into the sensor field from any point on the field boundary, and its moving trajectory in the sensor field can take arbitrary shapes. Complete area coverage can guarantee that every point of the moving trajectory within the sensor field can be detected. However, providing complete area coverage normally requires a great number of sensors to be deployed. Sometimes, it might not be necessary to detect a moving object at every point on its trajectory. Instead, it might be enough if the object can be detected at least by k distinct sensors before it penetrates through the sensor field. This requires a sensor network to provide *k-barrier coverage* over a belt field such that any *crossing path* intersects with the sensing areas of k distinct sensors. A crossing path is a moving trajectory (a continuous curve with arbitrary shape) with the start point on one side of the sensor field and the end point on the opposite side. This chapter discusses the barrier coverage, and the following questions are addressed in this chapter:

What is k-barrier coverage and how to provide k-barrier coverage?

This chapter first summarizes the definition and properties of k-barrier coverage and then discusses the algorithms to provide k barriers.

9.1 Sensor Barrier for Intrusion Detection

Intrusion detection and barrier coverage can find practical applications in real world. For example, the ExScal project [5] deployed more than 1000 sensors over a 1300 m $\times$ 300 m belt region to provide intrusion detection and barrier coverage. Kumar et al. [10, 11] introduce the following definition for k-barrier coverage and also investigate the properties of k-barrier coverage for belt-like regions. A belt region is defined by two parallel curves separated by a distance w. Let $d(x, l)$ denote the Euclidean distance between a point x and a curve l, that is, $d(x, l) = \min\{d(x, y) : y \in l\}$. Two curves l_1 and l_2 are said to be paralleled with

B. Wang, *Coverage Control in Sensor Networks,*
Computer Communications and Networks,
DOI 10.1007/978-1-84996-059-5_9, © Springer-Verlag London Limited 2010

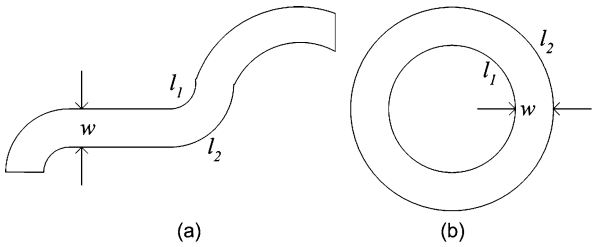

Fig. 9.1 Illustration of belt regions: (**a**) an open belt; (**b**) a closed belt

separation w if $d(x, l_2) = d(y, l_1) = w$ for all $x \in l_1$ and $y \in l_2$. Example belt regions are illustrated in Fig. 9.1. The following definitions are used for k-barrier coverage.

Definition 9.1 (Belt of width w) If l_1 and l_2 are two parallel curves with separation w, the region between l_1 and l_2 is referred to as a belt region of width w. The two curves l_1 and l_2 are the belt's parallel boundaries.

Definition 9.2 (Ordinary belt) A belt $\mathcal{B}$ is said to be an *ordinary belt* (with respect to a sensor network deployed over $\mathcal{B}$) if it satisfies the following condition: For two sensors within $\mathcal{B}$, if their sensing areas D_1 and D_2 have overlap, then $(D_1 \cup D_2) \cap \mathcal{B}$ is a connected subregion in $\mathcal{B}$.

Definition 9.3 (Crossing path and orthogonal crossing path) A path is said to be a *crossing path* if it crosses from one parallel boundary to the other. A crossing path is *orthogonal* if its length is equal to the belt width w.

Definition 9.4 (k-covered path) A path is *k-covered* if it intersects the sensing region of at least k distinct sensors.

Definition 9.5 (k-barrier coverage) A sensor network deployed over a belt region is said to provide k-barrier coverage if and only if all crossing paths through the belt are k-covered by the sensors.

For ease of presentation, a belt region is assumed from left to right, and two parallel boundaries are referred to as the top and the bottom boundary. Furthermore, intrusion movement is assumed to occur from top to bottom of the belt. Figure 9.2 illustrates these definitions in a rectangular belt. The cross paths in Fig. 9.2(a) cannot be 1-covered, and hence the network cannot provide barrier coverage. The network in Fig. 9.2(b) can provide 2-barrier coverage. It is seen that providing barrier coverage does not imply complete area coverage and barrier coverage normally requires less sensors than complete coverage.

Optimal Node Placement for k-Barrier Coverage If sensors are deterministically placed into a belt, what is the optimal configuration to provide k-barrier coverage? Kumar et al. [11] prove that the optimal configuration is to deploy k rows of

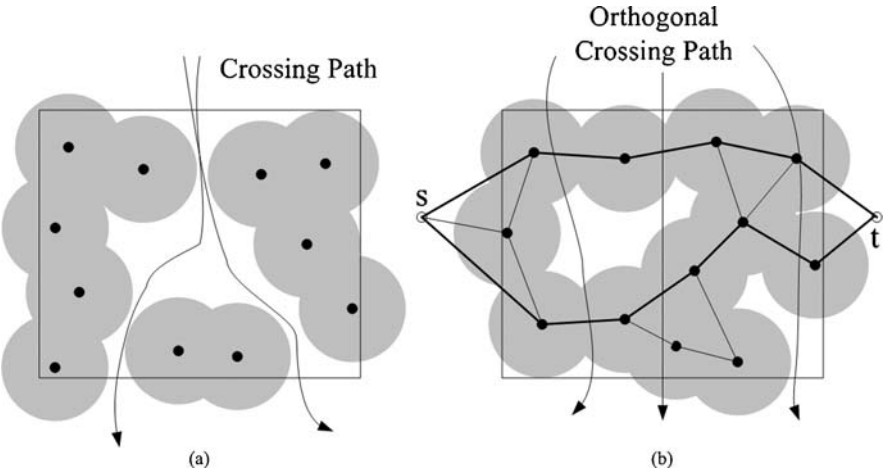

Fig. 9.2 Illustration of k-barrier coverage: (**a**) the network cannot provide barrier coverage; (**b**) the network can provide 2-barrier coverage

sensors along a shortest path (line or curve) across the length of the belt, where each path has consecutive sensors' sensing areas abutting each other.

Theorem 9.1 (Kumar et al. [11], Theorem 4.1) *Consider an open belt region. Let l denote the length of the shortest path across the length of the belt. Then the number of sensors necessary and sufficient to achieve k-barrier coverage is $k \times \lceil \frac{l}{2R_s} \rceil$, assuming that sensors are deployed to make the belt an ordinary belt.*

Existence Check of k-Barrier Coverage Another interesting question is, given a sensor network randomly deployed in a belt, how to determine whether the belt is k-barrier covered or not? Kumar et al. [10] show that this question cannot be locally answered by individual sensors. That is, individual sensor cannot decide, based only on the information from its neighbors, whether the belt is k-barrier covered. Kumar et al. [10] propose to use a global *coverage graph* to check k-barrier coverage for an open belt.

Definition 9.6 (Coverage graph, CG) A coverage graph $CG = (V, E)$ of a sensor network is constructed as follows. The vertex set V corresponds to all sensor nodes. In addition, it has two virtual nodes, s and t, corresponding to the left and right boundaries. An edge exists between two nodes if their sensing regions overlap in the belt region. An edge exists between a node and s (or t) if the sensing region of the node overlaps with the left boundary (or right boundary) of the belt.

Theorem 9.2 (Kumar et al. [11], Theorem 4.1) *A sensor network N is deployed over an open and ordinary belt region $\mathcal{B}$. The belt region $\mathcal{B}$ is k-barrier covered by the sensor network N iff there exist k node-disjoint paths between the two virtual nodes s and t in the coverage graph CG.*

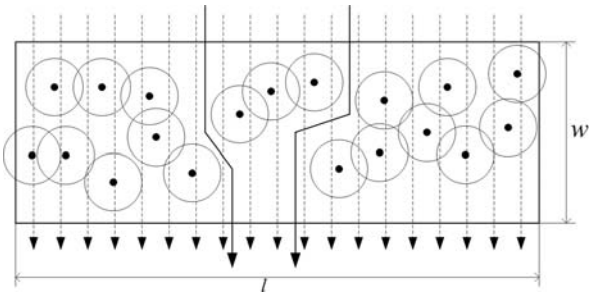

Fig. 9.3 Illustration of weak barrier coverage which only guarantees detecting intruders moving on congruent crossing paths

For example, in Fig. 9.2(b), a CG is constructed for the deployed sensor nodes, and two node-disjoint paths (the bold lines) between s and t exist. Hence, according to Theorem 9.2, the belt is 2-barrier covered. Some existing algorithms can be used to find k node-disjoint paths with $O(k^2|V|)$ complexity [15]. For k-barrier coverage in a closed belt, it is still considered as an open problem to design an efficient algorithm to determine whether a closed belt is k-barrier covered [11]. For some specific closed belts such as donut-shaped belts (cf. Fig. 9.1(b)), Kumar et al. propose to check k node-disjoint essential cycle in the corresponding coverage graph for k-barrier coverage [10].

Critical Density for Providing k-Barrier Coverage If sensors are randomly deployed within an open belt, what is the minimum number of sensors required to provide k-barrier coverage? This problem of critical density for k-barrier coverage has been studied in [1, 11, 13, 14]. In what follows, we briefly introduce the analysis results for an open rectangular belt and the sensing disk coverage model.

Instead of considering k-barrier coverage for *every* crossing path, Kumar et al. [11] consider the k-barrier coverage for only a subset of *orthogonal* crossing paths. Obviously, this only provides *weak* barrier coverage since the intrusion trajectory may not follow an orthogonal crossing path. As shown in Fig. 9.3, all orthogonal crossing paths are 1-barrier covered. However, the belt is not 1-barrier covered as there some crossing paths (the bold lines) are not been 1-barrier covered. Let l and w denote the length and width of a rectangular belt. Kumar et al. [11] normalize l and w as $l \times w = 1$. The network deployment is assumed to be driven by a Poisson point process with density λ, and the sensing radius is R_s for each sensor. Furthermore, it is assumed that each sensor uses a random independent sleep (RIS) scheduling and is active with probability p. The sensor network hence is represented by $N(\lambda, p, R_s)$. The function $\phi(\lambda p)$ denotes an arbitrary, slowly and monotonically increasing function that goes to infinity and $\phi(\lambda p) = o(\log\log(\lambda p))$. The following theorem establishes a sufficient condition for the k-coverage *with high probability* (*w.h.p.*) of all orthogonal crossing paths.

Theorem 9.3 (Kumar et al. [11], Theorem 6.1) *Let $N(\lambda, p, R_s)$ be a Poisson-distributed sensor network over an $l \times w$ ($l \times w = 1$) belt region. If $c(l) = \frac{2\lambda p R_s}{l \log(\lambda p)}$*

satisfies

$$c(l) \geq 1 + \frac{\phi(\lambda p) + (k-1)\log\log(\lambda p)}{\log(\lambda p)}$$

for sufficiently large l, then all the orthogonal crossing paths in the belt region are k-barrier covered w.h.p. as $l \to \infty$.

Kumar et al. [11] also provide a sufficient condition for the existence of an uncovered orthogonal crossing path in a rectangular belt region.

Theorem 9.4 (Kumar et al. [11], Theorem 6.2) *Let $N(\lambda, p, R_s)$ be a Poisson-distributed sensor network over an $l \times w$ ($l \times w = 1$) belt region. If $c(l) = \frac{2\lambda p R_s}{l\log(\lambda p)}$ satisfies*

$$c(l) \leq 1 - \frac{\phi(\lambda p) + \log\log(\lambda p)}{\log(\lambda p)}$$

for sufficiently large l, then there exists an non-k-covered orthogonal crossing path in the region w.h.p. as $l \to \infty$.

Liu et al. [13] study the critical conditions for *strong* barrier coverage. A sensor network is said to be strongly covered if the probability that *any* crossing path is k-covered equals 1 with high probability. They apply the results from the percolation theory (e.g., [7]) and derive critical conditions for *strong* barrier coverage as follows.

Theorem 9.5 (Liu et al. [13], Theorem 1) *Consider a sensor network deployed on a two-dimensional rectangular belt region $\mathcal{B} \doteq [0, l] \times [0, w(l)]$, where sensors are distributed according to a Poisson point process with density n.*

- *If $w(l) = \Omega(\log(l))$, the network is strongly barrier covered w.h.p. when the sensor density reaches a certain value. There exists a positive constant β such that w.h.p. there exists $\beta w(l)$ disjoint horizontal sensor barriers crossing the belt region.*
- *If $w(l) = o(\log(l))$, the network has no strong barrier coverage w.h.p. regardless what the sensor density is in the underlying sensor network. That is, w.h.p. there exist crossing paths that an intruder can cross the belt without being detected.*

It follows from this theorem that the existence and strength of the strong barrier coverage depends on the width-to-length ratio of the belt region. The critical condition for strong barrier coverage to exist is when the width of the strip becomes asymptotically larger than the logarithm of the length, i.e., $w(l) = \Omega(\log(l))$. The number of horizontally connected sensor barriers is proportional to the width of the strip by a constant factor. Their analysis approach has been extended to a more practical scenario where all sensor nodes are randomly deployed in a thin belt region (the width of the belt is not too much larger than the sensing range) [14]. As nodes are confined within such a thin belt, their offsets to a single barrier are much smaller

than that generated by a random deployment over a rectangle field, and the critical density for establishing a barrier can be significantly reduced.

Balister et al. [1] derive critical density estimates for achieving barrier coverage in finite-size thin belt regions. Their analysis is based on the connectivity property of the coverage graph defined in Definition 9.6. Let $G_{w,r,\lambda}(0,l)$ denote the coverage graph for a rectangular belt $[0,l] \times [0,w]$, where λ is the Poisson deployment density, and r the sensing radius. They define a *break* as a disruption of the connectivity of the coverage graph. Obviously, if there exists a break, then there also exist crossing paths in the belt that do not intersect with any sensor's sensing area. Break intensity informally denotes how frequently one expects a loss in $s-t$ connectivity when traversing the length of the belt. They show that the breaks occur with an approximately Poisson distribution. Then the probability of no barrier coverage equals the probability that $G_{w,r,\lambda}(0,l)$ is not $s-t$ connected,

$$\Pr\big[G_{w,r,\lambda}(0,l) \text{ is not } s-t \text{ connected}\big] \approx 1-\exp(-lI_{w,r,\lambda}) \approx lI_{w,r,\lambda},$$

where $I_{w,r,\lambda}$ is the break intensity, and $lI_{w,r,l} \leq 1$. For $r \geq 3$ and $wr^{-\frac{1}{3}} \geq 2$, the break intensity can be approximated as follows:

$$I_{w,r,\lambda} \approx \sqrt{\lambda}\exp(-\alpha_r w - \beta_r),$$

where $\alpha_r = \sqrt{\lambda}(r - 1.12794r^{-\frac{1}{3}} - 0.20r^{-\frac{5}{3}})$ and $\beta_r = \sqrt{\lambda}(-\frac{1}{3}\log r + 1.05116 + 0.27r^{-\frac{4}{3}})$. The analysis in [1] does not rely on asymptotic analysis, and hence the analysis results have more practical usage for finite-size belt.

9.2 Sensor Scheduling for Barrier Construction

In random sensor deployments, sensor activity scheduling can be used to activate sensors alternatively to form barriers, which helps to prolong the network lifetime. This is similar to the problem of scheduling sensor activity for area coverage. However, as the objective is different, providing barrier coverage other than providing area coverage, the solution approaches are different. Several sensor activity scheduling algorithms for barrier coverage have been proposed in [3, 4, 11–13, 17]. The simplest scheduling algorithm again is the random independent sleeping (RIS) scheme where each sensor self activates itself for some time with some probability p and independently of other sensors. In the RIS approach, the barrier coverage can only be guaranteed probabilistically. In order to achieve high probability of achieving barrier coverage, more sensor nodes or small activation probability p are used. The relation between the activation probability p, the network size, the length of the belt, and the weak barrier coverage has been studied in [11] and summarized in Theorem 9.3 in the previous section.

Kumar et al. [12] propose two algorithms, called Stint and Prahari, to optimally schedule sensors' activity. The optimality means that the two algorithms can achieve the upper bound of the network lifetime. The Stint algorithm is for homogeneous networks, and the Prahari algorithm is for heterogeneous networks.

- For a homogeneous network where all sensors have the same lifetime, the maximum time for such a network to provide k-barrier coverage is at most m/k, where m is the maximum number of node-disjoint paths connecting the two virtual nodes s and t in the coverage graph (cf. Definition 9.6). The Stint algorithm first activates $l = \lfloor m/k \rfloor - 1$ node-disjoint paths in sequence until all of them exhaust their lifetimes. The remaining $r = m - l \times k$ disjoint paths are arranged in $f = r/\gcd(r, k)$ sets of k disjoint paths, where $\gcd(r, k)$ returns the *greatest common divisor* for r and k. Each of these f sets of paths is kept active for $\gcd(r, k)/k$ of the total lifetime of a sensor. In this way, the network provides k-barrier coverage for $l + r/k = m/k$ time units and hence achieves lifetime upper bound.
- For a heterogeneous network where sensors have different lifetimes, the Prahari algorithm first constructs a *coverage graph with lifetime* CG_L by adding the sensor lifetime as the capacity of the vertex in the coverage graph. The network lifetime for k-barrier coverage is determined by the maximal value of the *composite k-flow* of the graph CG_L. A composite k-flow consists of a set of *basic k-flows* each of which has a value a_i and can provide k-barrier coverage. If the maximal value of the composite k-flow in CG_L is $\hat{f}$, then the maximum time for such a network to provide k-barrier coverage is $\hat{f}/k$. The Prahari algorithm applies the MEM algorithm from [9] to determine $\hat{f}$ and the flow network F_{MEM}. If the in-degree and out-degree of all nodes except the virtual nodes s and t is 1, then the flow network F_{MEM} is decomposed into $m > k$ node-disjoint path flows, and a machine scheduling algorithm is used to schedule the m paths to achieve a lifetime of $\hat{f}/k$. If the in-degree and out-degree of some nodes in F_{MEM} is more than 2, then the Prahari algorithms invokes the SEM algorithm form [9] to decompose the flow network and then merges into identical basic k-flows and schedules these basic k-flows one by one to achieve the maximum lifetime $\hat{f}/k$.

The Stint and Prahari algorithms compute sensor barriers which guarantee that any crossing path within the belt can be detected, even if the crossing path spans the whole length of the belt. However, they are centralized algorithms and might not be scalable for large-scale networks.

In [3], Chen et al. define L-local k-barrier coverage and propose Localized Barrier Coverage Protocol (LBCP) to let each sensor locally schedule its own activity state only based on the information of its neighbors. Chen et al. refer to the barrier coverage defined in [11] as *global* barrier coverage as it guarantees intrusion detection for *any* crossing path within the belt. As illustrated in Fig. 9.4, the belt is not globally barrier covered as there exists a long uncovered crossing path. Chen et al. argue that movements are likely to follow a shorter path when crossing a belt region, and hence the trajectory is likely to be bounded within a slice of length L, called L-zone. They then define L-local barrier coverage to guarantee the detection of all crossing paths whose trajectory is confined to such a L-zone.

Definition 9.7 (L-local k-barrier coverage) For $L > 0$ and $k = 1, 2, \ldots$, a belt region is said to be L-local k-barrier covered if every L-zone in the region is k-barrier covered.

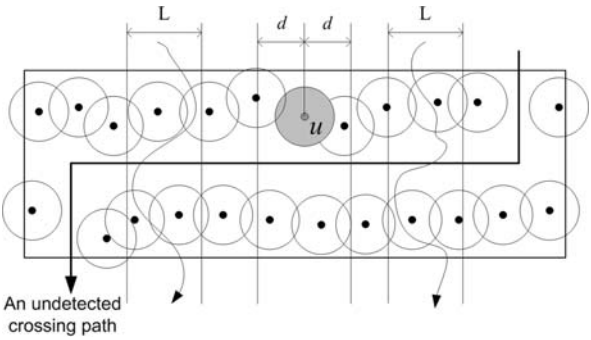

Fig. 9.4 Illustration of global barrier coverage and L-local barrier coverage

The belt L-Local k-barrier coverage is related to the k-barrier coverage of each sensor's $2d$-zone. A $2d$-zone(u) of a node u is defined as an L-zone with $L = 2d$, and the orthogonal crossing line passing through u divides the L-zone into two sections of equal length (each of length d). The following theorem indicates when, and for what value of L, one can conclude that a rectangular belt is L-local k-barrier covered.

Theorem 9.6 (Chen et al. [3], Theorem 4.1) *Consider a rectangular belt with at least one active sensor node. If $2d$-zone(u) for every active node u is k-barrier covered for some $d > r$, where r is the sensing radius of the sensing disk, then the entire belt is L-local k-barrier covered, with $L = \max\{2d - 2r, d + r\}$.*

According to this theorem, each node needs only to determine whether its own $2d$-zone is k-barrier covered or not.

In the LBCP scheme, each node u only needs to communicate with node v within the distance of $D_u = \max\{d(u, v) : v \in 2d\text{-zone}(u) \text{ or } u \in 2d\text{-zone}(v)\}$. After sleeping for a random time period, a waking-up node performs the following steps to decide whether it should be active or sleep for another period of time.

1. A waking-up node u broadcasts a *Query_W* message in the rang of D_u.
2. When an active node v receives *Query_W* from u, if u is in $2d$-zone(v) and v is not k-barrier covered, then v replies with a *Required_W* message. Otherwise, v replies a *Not_Required_W* message.
3. If u receives *Required_W*, it becomes active. If u does not receive any *Required_W* or *Not_Required_W*, which means there are no active nodes in $2d$-zone(u), u also becomes active. Otherwise, u goes back to sleep.
4. If u decides to go back to sleep, u sleeps until T time units or until the first active node in the range of D_u is expected to die, whichever occurs earlier.
5. If u decides to become active, u broadcasts in the range of D_u a *Decision_Active* message.

An active node u can also go back to sleep if for every active node v such that $u \in 2d$-zone(v), $2d$-zone(v) will be k-barrier covered without u. This is again based on the activity information of its D_u neighbors by broadcasting some query messages.

A dual question in L-local k-barrier coverage is to find, for a given randomly deployed sensor network, what is the maximum L that the belt can be L-local k-barrier covered. In [4], Chen et al. define it as the quality of L-local k-barrier coverage.

Definition 9.8 (Quality of k-barrier coverage, Q_k) The quality of sensor deployment for k-barrier coverage, denoted by Q_k, is defined as the maximum L such that the belt is L-local k-barrier covered; i.e., $Q_k = \max\{L :$ the belt is L-local k-barrier covered$\}$. If there is no such L (i.e., if the belt is not even 0-local k-barrier covered), then define $Q_k = -1$.

The questions of how to determine whether $Q_k = -1$ and how to determine Q_k have been discussed by Chen et al., and the interested readers can refer to [4] for details.

9.3 Sensor Barrier with Mobile Nodes

When sensors are randomly deployed, the number of nodes to be deployed is normally much higher than the optimum to provide barrier coverage. On the other hand, deterministic node placement can greatly reduce the number of nodes as they can be placed on the optimal positions to form a barrier. Another method is to use mobile nodes. Even if mobile nodes are randomly deployed at first, they can move to desired locations to form a barrier for a sensor field. Several node movement strategies that specify how to efficiently construct a barrier via the nodes' movement have been studied in [2, 6, 8, 16]. The objectives of such a movement strategy include two aspects: one is to determine the optimal locations in a barrier, and another is to determine which node should move to which location so that some moving cost can be minimized.

In [2], two types of moving cost are identified. One is to minimize the maximum moving distance among all nodes, and another is to minimize the total moving distance over all nodes. The barrier is required to be along the perimeter of a circle or a convex polygon. The optimal pattern of such a barrier coverage is to put sensors with equidistance for any two clockwise successive nodes along the perimeter of the circle or the polygon. This actually forms an n-regular polygon (regular n-gon) inscribed in the sensor field. An example is illustrated in Fig. 9.5, where eight mobile nodes are randomly deployed within a circle, and the optimal locations are the vertices of an inscribed octagon. Although the optimal barrier pattern is a regular n-gon inscribed with the circle, the determination of the optimal target locations may not be trivial. This is because the orientation of the n-gon can take any angle.

Bhattacharya et al. [2] propose several algorithms to determine the optimal locations and the movement scheme. Let $\mathcal{C}$ denote the perimeter of the circle. Let A_0 denote the location of the disk center, and let A_i denote the location of a sensor. If $A_0 \neq A_i$, then the radius $\overline{A_0 A_i}$ intersects $\mathcal{C}$ with two points; let B_i denote the one on $\mathcal{C}$ with the smaller distance between A_i and $\mathcal{C}$. If $A_i = A_0$, then an arbitrary point on $\mathcal{C}$ is selected to be B_i. The vertices of a regular n-gon can be parameterized by

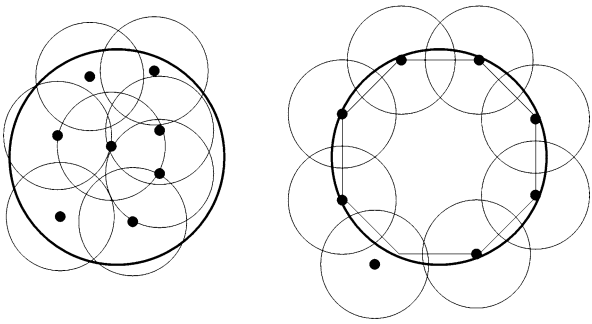

Fig. 9.5 Illustration of providing barrier coverage for a circle with mobile sensor nodes: (**a**) initial deployment; (**b**) after movement

$\phi + \frac{2(j-1)\pi}{n}$ for $j = 1, 2, \ldots, n$ and $0 \le \phi < \frac{2\pi}{n}$. For a fixed ϕ, let B'_i denote these vertices on $\mathcal{C}$ with an ascending order of their angels. By arranging B_i also in an ascending order according to their angles (the angle of vector $\overrightarrow{A_0 B_i}$), then the target location for sensor located at B_i is B'_i, $i = 1, 2, \ldots, n$. The movement algorithm for minimizing total moving distance consists of the following three steps:

Step 1: Compute B_i for each sensor A_i, $1 \le i \le n$.
Step 2: Compute the target locations B'_i of a regular n-gon for B_i.
Step 3: Move A_i to B'_i, $1 \le i \le n$, and compute $S_n^1 = \sum_{i=1}^{n} d(A_i, B'_i)$.

Bhattacharya et al. [2] show that this algorithm can be implemented in $O(n^2)$ time, and its approximation ratio is no more than $\pi + 1$.

Shen et al. [16] propose to apply the concept of *virtual forces*[1] to move nodes to form a barrier in a rectangular belt $[0, L] \times [0, W]$. The virtual force consists of both repulsive force, which repels two sensors apart, and attractive force, which pulls two sensors together. Let $\mathcal{N}_i$ denote the set of neighbors of node i. The repulse force between neighboring sensor s_i and s_j is computed as

$$\overrightarrow{F_r(i,j)} = \frac{a}{(x_i - x_j)}, \quad s_j \in \mathcal{N}_i, \quad \text{and} \quad \overrightarrow{F_r(i,j)} = 0, \quad s_j \notin \mathcal{N}_i,$$

and the attract force is computed as

$$\overrightarrow{F_a(i,j)} = \frac{b}{(y_j - x_i)}, \quad s_j \in \mathcal{N}_i, \quad \text{and} \quad \overrightarrow{F_a(i,j)} = 0, \quad s_j \notin \mathcal{N}_i,$$

where a and b are two positive constants. It is easy to see that the repulsive forces are to incite nodes to a uniform distribution and the attractive forces are to gather sensors into a same horizon. The left and right boundary also exerts repulsive forces on a sensor if the distance between this sensor and the boundary is less than the

[1] Virtual force for node movement has been discussed in Chap. 8, Sect. 8.2.2, for providing area coverage.

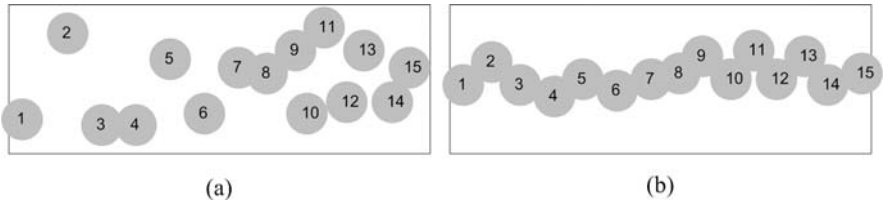

Fig. 9.6 Illustration of a barrier formed by using virtual forces: (**a**) initial deployment; (**b**) after movement

communication range R_c:

$$\overrightarrow{F_r(i,B)} = \begin{cases} -\frac{1}{x_i}, & x_i < R_c, \\ 0, & R_c < x_i \leq (L - R_c), \\ \frac{1}{x_i - L}, & x_i > L - R_c. \end{cases}$$

The total repulsive force on s_i is the vector addition of all repulsive forces from its neighbors and the boundary, and the total attractive force on s_i is the vector addition of all attractive forces from its neighbors. Figure 9.6 illustrates an example of using virtual forces to relocate mobile nodes. Since each node only needs to exchange information with its neighbors, this movement algorithm is localized and distributed and can also be applied to large-scale networks.

References

1. Balister, P., Bollobas, B., Sarkar, A., Kumar, S.: Reliable density estimates for coverage and connectivity in thin strips of finite length. In: ACM the 13th International Conference on Mobile Computing and Networking (MobiCom), pp. 75–86 (2007)
2. Bhattacharya, B., Burmester, B., Hu, Y., Kranakis, E., Shi, Q., Wiese, A.: Optimal movement of mobile sensors for barrier coverage of a planar region. In: The 2nd International Conference on Combinatorial Optimization and Applications (COCOA), also in LNCS, vol. 5165, pp. 103–115 (2008)
3. Chen, A., Kumar, S., Lai, T.H.: Designing localized algorithms for barrier coverage. In: ACM the 13th International Conference on Mobile Computing and Networking (MobiCom), pp. 63–74 (2007)
4. Chen, A., Lai, T.H., Xuan, D.: Measuring and guaranteeing quality of barrier-coverage in wireless sensor networks. In: ACM the 9th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 421–430 (2008)
5. ExScal: Extreme scale wireless sensor networking. http://cast.cse.ohio-state.edu/exscal (2008)
6. Flocchinia, P., Prencipeb, G., Santoro, N.: Self-deployment of mobile sensors on a ring. Journal of Theoretical Computer Science (Elsevier) **402**(1), 67–80 (2008)
7. Grimmett, G.: Percolation. Springer, Berlin (1999)
8. Jafari, M.A., Liu, J., Golmohammadi, D.: Network flow formulation of optimal perimeter sensory coverage problem. European Journal of Operational Research (Elsevier) **197**(1), 77–83 (2009)
9. Kishimoto, W.: A method for obtaining the maximum multiroute flows in a network. Networks **27**(4), 279–291 (1996)

10. Kumar, S., Lai, T.H., Arora, A.: Barrier coverage with wireless sensors. In: ACM International Conference on Mobile Computing and Networking (MobiCom), pp. 284–298 (2005)
11. Kumar, S., Lai, T.H., Arora, A.: Barrier coverage with wireless sensors. Wireless Networks **13**(6), 817–834 (2007)
12. Kumar, S., Lai, T.H., Posner, M.E., Sinha, P.: Optimal sleep-wakeup algorithms for barriers of wireless sensors. In: IEEE the 4th International Conference on Broadband Communications, Networks, and Systems (Broadnets), pp. 1–10 (2007)
13. Liu, B., Dousse, O., Wang, J., Saipulla, A.: Strong barrier coverage of wireless sensor networks. In: ACM the 9th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 411–419 (2008)
14. Saipulla, A., Westphal, C., Liu, B., Wang, J.: Barrier coverage of line-based deployed wireless sensor networks. In: IEEE Infocom, pp. 127–135 (2009)
15. Schrijver, A.: Combinatorial Optimization. Springer, Berlin (2003)
16. Shen, C., Cheng, W., Liao, X., Peng, S.: Barrier coverage with mobile sensors. In: IEEE International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN), pp. 99–104 (2008)
17. Yang, G., Qiao, D.: Barrier information coverage with wireless sensors. In: IEEE Infocom, pp. 918–926 (2009)

Chapter 10
Find Penetration Paths

Abstract The coverage problem of finding penetration paths is rooted in the intrusion detection and tracking applications. A *penetration path* is a crossing path (a continuous curve with arbitrary shape), which enters the sensor field from one side and leaves the sensor field from the other side. For example, given the knowledge of the network deployment, an intruder would like to find a safest penetration path such that when it moves along the path, it is most unlikely to be detected and tracked. From the viewpoint of a defender, a safest penetration path to the intruder is its weakest path. A defender needs to identify such worst penetration paths and takes some measure to improve the network monitoring performance. The objective of finding penetration paths is to identify *one* crossing path with *every* point on it whose coverage measure satisfies a predefined coverage requirement. This is different from the problem of building intrusion barriers, which is mainly to guarantee that *some* points of *every* crossing path should meet certain coverage requirements. This chapter discusses the coverage problem of finding penetration paths, and the following questions are addressed in this chapter.

What are penetration paths and how to find a penetration path?

The definition and the property of a penetration path are dependent on the coverage model and coverage requirements. This chapter introduces some typical types of penetration path and presents the algorithms for finding penetration paths.

10.1 Maximal Breach Path

Megerian et al. [18, 19] might be the first to study the penetration path problem. They apply the Euclidean distance between a point and its closest sensor node as the coverage measure for a point and introduce the following *maximum breach path* problem. Let $\mathcal{S}$ denote the set of sensor nodes. Given an initial location I and a final location F, let P denote a path connecting I and F.

Definition 10.1 (Path breach) The *breach* of a path P connecting I and F is defined as the minimum Euclidean distance from P to any sensor in $\mathcal{S}$.

B. Wang, *Coverage Control in Sensor Networks,*
Computer Communications and Networks,
DOI 10.1007/978-1-84996-059-5_10,

For every point on P, we measure the Euclidean distance between the point and its nearest sensor and then take the minimum among these distances.

Definition 10.2 (Maximal breach path) Among all the paths connecting I and F, the one with the maximum breach value is called a maximal breach path, denoted by P_B.

From the viewpoint of an intruder, P_B is the safest path, since the distance from the nearest sensor along P_B is maximized. But for the defender, this is a worst path since the chance of detecting an intruder is minimized. This is called *worst coverage* in [18].

There are infinitely many paths connecting I and F, and exhaustive search for the maximal breach path is not possible. Megerian et al. [18] apply the Voronoi diagram to provide a geometric division of the sensor field and argue that at least one maximal breach path lies on the Voronoi edges.

Theorem 10.1 (Megerian et al. [18], Theorem 1) *At least one maximal breach path must lie on the line segments of the bounded Voronoi diagram formed by the locations of the sensors in $\mathcal{S}$.*

We refer the reader to Appendix A for a brief introduction of Voronoi diagrams. According to this theorem, one can restrict the search for P_B only on Voronoi edges. It is worth noting that the maximal breach path is not unique. There may have many other paths qualified as maximal breach paths.

Megerian et al. [18] provide a centralized algorithm to find a P_B. The algorithm performs binary-search and breadth-first-search to find a P_B on the graph induced from the Voronoi diagram.

1. Convert the Voronoi diagram to an undirected and weighted graph G where the weight of an edge, w, is set as the minimum distance between this edge and the closest sensor.
2. Initialize the breach weight W as $W = w_{\min} + w_{\max}$, where $w_{\min}$ and $w_{\max}$ are the minimum and maximum edge weight in G, respectively, and initialize the search range as $SR = (w_{\max} - w_{\min})/2$.
3. Construct a new graph G' from G by dropping all edges with weight smaller than W.
4. Apply breadth-first-search to find a path on G' between I and F. If a path is not found, then set $W \leftarrow W - SR/2$. If a path is found, then set $W \leftarrow W + SR/2$.
5. Set $SR \leftarrow SR/2$. If $SR < SR\_Threshold$, then the algorithm terminates; otherwise go to (3).

Pruning edges and the breadth-first-search run in time linear in the number of the nodes and edges of the graph G. The worst case of the binary search requires $\log C$ iterations, where $C = w_{\max} - w_{\min}$. Then the overall complexity of this algorithm is $O(N \log C)$. Figure 10.1, taken from [18], illustrates a weighted Voronoi diagram and shows the search result, a maximal breach path (the bold doted line).

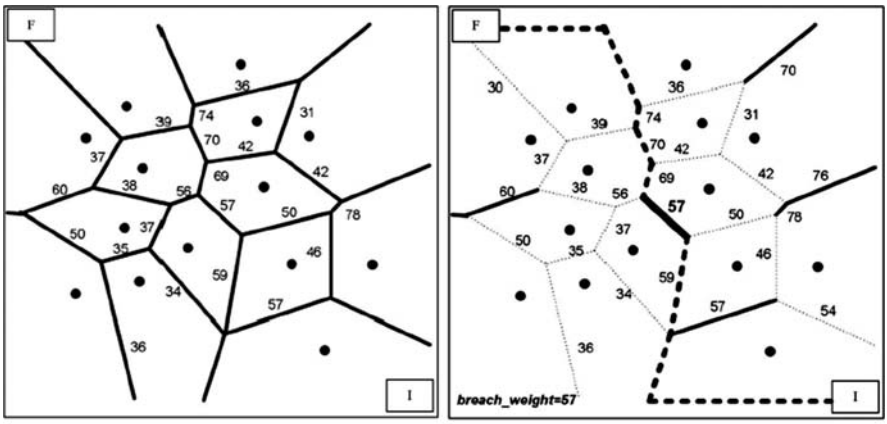

Fig. 10.1 Illustration of finding maximal breach paths: (**a**) the weighted Voronoi diagram; and (**b**) the maximal breach path (reproduced from [18], ©2005, IEEE)

After Megerian et al., many other researchers have also studied such breach path problems and proposed improved algorithms to find a maximal breach path or to minimize path breach [1, 7–9, 12–14, 21]. We list a few in what follows. Mehta et al. [21] argue that a P_B can also be found on the maximum spanning tree of the induced Voronoi graph $VG(\mathcal{S})$. Mehta et al. [21] also consider a redundant breach problem where the coverage measure of a point is defined as the Euclidean distance between this point and its kth closest sensor. This corresponds to the scenario that an intruder is detected only if it is simultaneously near k sensors. To find such a redundant maximal breach path, the kth nearest sensor Voronoi diagram is used instead of the ordinary Voronoi diagram.

In the above formulation of the maximum breach path problem, each node is implicitly assumed to have a sensing capability linear in the distance. Mehta et al. [21] consider the case that all sensors have limited sensing capability, e.g., the sensing disk model. If the network density is not very high, then there may exist some regions in the sensor field that are not covered by any of sensing disks, i.e., the coverage holes. An intruder will not be detected in such coverage holes. Therefore, it can take a shortcut when moving from one Voronoi vertex to another as long as the shortcut is within coverage holes. Therefore, the length of P_B in such a case can be further shortened.

The Voronoi diagram cannot be constructed locally or even efficiently in a distributed way, which limits the usage of the above centralized P_B search algorithm. Huang et al. [14] propose an *aberrant Voronoi graph* to approximate the ordinal Voronoi diagram, which can be constructed locally. Each node derives its aberrant Voronoi region as the intersection of its sensing area and the ordinal Voronoi cell constructed based only on its one-hop neighbors. The aberrant Voronoi graph is the union of such aberrant Voronoi regions of all nodes, and its edges may consist of both lines and curves. A distributed Bellman–Ford algorithm is then applied in such

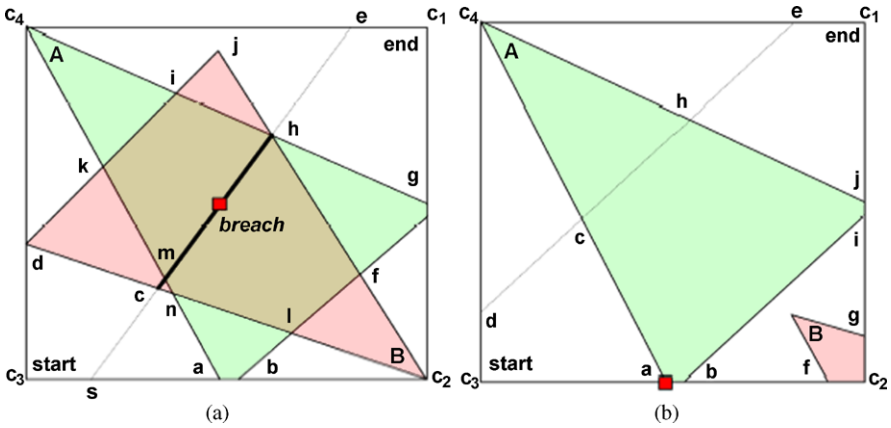

Fig. 10.2 Illustration of constructing field-of-view Voronoi graph (reproduced from [1], ©2006, IEEE)

aberrant Voronoi graph to find the shortest path connecting I and F as the maximal breach path.

Adriaens et al. [1] consider the maximal breach path problem for directional sensors. Unlike an isotropic sensor, the sensing capability of a directional sensor is confined within a sector, called *field of view* (FOV). The ordinary Voronoi diagram is based on the isotropic sensing model. Adriaens et al. [1] propose to construct a FOV-Voronoi graph and find a P_B on the edges of this graph. The FOV-Voronoi graph is constructed by intersecting the edges of all the field-of-view regions with the Voronoi edges in each resulting polygon that is observed by more than one sensor. Only the sensors observing a shared polygon are used in the construction of the Voronoi edges in that polygon. Figure 10.2, taken from [1], illustrates the FOV-Voronoi graph for two directional sensors. The FOV of a sensor is represented by an isoceles triangle. The maximal breach path is $\langle c_3, d, c, m, h, g, c_1\rangle$ and $\langle c_3, a, b, i, j, c_1\rangle$, respectively.

10.2 Maximal Support Path

Megerian et al. [18] propose the problem of finding a *maximal support path*, which also uses the Euclidean distance between a point and its closest sensor as the coverage measure. Similar to the worst-case (breach) coverage formulation, the support of a path is defined for a given initial location I, a given final location F, and for the sensor set S, as follows.

Definition 10.3 (Path support) The *support* of a path P connecting I and F is defined as the maximum Euclidean distance from P to the closest sensor in $\mathcal{S}$.

For every point on P, we measure the Euclidean distance between the point and its nearest sensor and then take the maximum among these distances.

Definition 10.4 (Maximal support path) Among all the paths connecting I and F, the one with the lowest support value is called a maximal support path, denoted by P_S.

From the viewpoint of an intruder, this is the worst possible path to take as the chances of being detected are maximized. But for the defender, this is the best path along which the surveillance quality is the highest. This is called *best coverage* in [18].

To reduce search space, Megerian et al. [18] apply the Delaunay triangulation to provide a geometric division of the sensor field and argue that at least one maximal support path lies on the edges of the Delaunay triangulation.

Theorem 10.2 (Megerian et al. [18], Theorem 2) *At least one maximal breach path must lie on the line segments of Delaunay triangulation* (*with the exceptions of the start and end points connecting* P_s *to* I *and* F).

We refer the reader to Appendix A for a brief introduction of Delaunay triangulation. According to this theorem, one can restrict the search for P_S only on the edges of the Delaunay triangulation.

The algorithm used for searching P_B in the previous section can also be used to find P_S with the following changes.

1. The Voronoi diagram is replaced by the Delaunay triangulation as the underlying geometric structure.
2. Each edge in graph is assigned a *support weight* equal to the largest distance from the corresponding line segment in the Delaunay triangulation to the closest sensor.
3. The search parameter W is now called *support weight*, and the search is conducted in a way that W is minimized.

Again, it is worth noting that the maximal support path is not unique. There may be many other paths qualified as maximal support paths. Figure 10.3, taken from [18], illustrates a P_S on the Delaunay triangulation and both P_B and P_S in a same network.

In general, the Delaunay triangulation cannot be constructed efficiently in a distributed way, which limits the usage of the centralized P_B and P_S search algorithm. However, besides the Delaunay triangulation, other geometric structures can also be used to find a maximal support path. Li et al. [15] prove that a maximal support path can also be found from the edges of the Gabriel graph or from the edges of the *relative neighborhood graph* (RNG). An RNG can be constructed locally. Let lune(u, v) denote the intersection area of the two disks with radius $d(u, v)$ and centered at u and v, respectively. A node u obtains its neighbor set $\mathcal{N}(u)$ from message broadcasts, and u adds an edge uv to the RNG$(\mathcal{S})$ iff the lune(u, v) does not contain

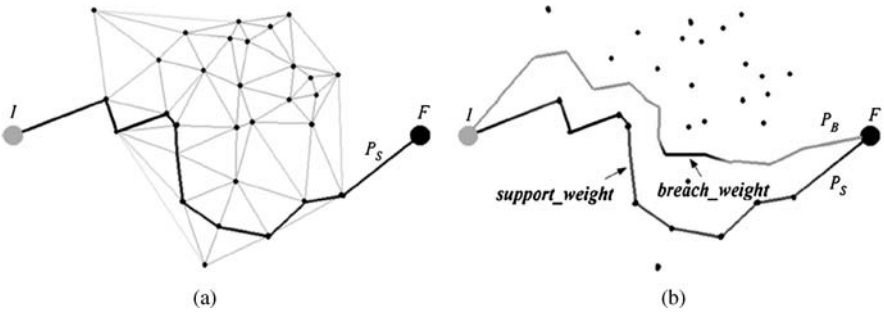

Fig. 10.3 (**a**) Illustration of a maximal support path on the Delaunay triangulation; and (**b**) illustration of a maximal support path and a maximal breach path in a same network (reproduced from [18], ©2005, IEEE)

any other nodes (except v) in $\mathcal{N}(u)$. After constructing an RNG, the search of a P_S connecting I and F then can be converted to finding a *shortest path* from I and F in RNG($\mathcal{S}$), which can be done in a distributed way, e.g., the distributed Bellman–Ford algorithm [5].

10.3 Exposure Path

Exposure path is another type of penetration path which measures how well a sensing field is covered in terms of the *expected ability* to detect a moving target. The higher the exposure, the better the coverage that the network can provide. Similar to the breach path and support path, a distance-dependent sensing function is used as the *coverage measure* to define the exposure of a point [17, 20, 25]. In the maximal breach path or the maximal support path problem, the breach or the support of a path is defined as the extremum value of the coverage measure of all points on the path. But in the exposure path problem, the exposure of a path is defined as the average of the coverage measure of all points on the path.

Megerian et al. [17] define the sensibility of a point p by a sensor s as inversely proportional to their distance:

$$S(s, p) = \frac{A}{[d(s, p)]^{\alpha}},$$

where the constants A and α are sensor technology-dependent parameters. Given this sensor coverage function, two models can be used to measure the sensing field intensity exerted by the sensor field F to a point p. The *closest sensor field intensity* is defined as $I_C(F, p) = S(s_{\min}, p)$, where $s_{\min}$ is the sensor closest to point p. The *all sensor field intensity* is defined as $I_A(F, p) = \sum_{i=1}^{n} S(s_i, p)$, where every active sensor s_i contributes a certain amount of sensitivity to the point p depending on its distance to the point.

Suppose that an intruder is moving in the field F from point $p(t_1)$ to point $p(t_2)$ along the path $p(t)$. The exposure of this path is defined as follows.

Definition 10.5 (Path exposure) The exposure for an intruder in the sensor field F during the interval $[t_1, t_2]$ along the path $p(t)$ is defined as

$$E\big(p(t), t_1, t_2\big) = \int_{t_1}^{t_2} I\big(F, p(t)\big) \left| \frac{dp(t)}{dt} \right| dt,$$

where the sensor field intensity $I(F, p(t))$ can be either $I_A(F, p(t))$ or $I_C(F, p(t))$, and $|dp(t)/dt|$ is the element of arc length. For example, if $p(t) = (x(t), y(t))$, then $|\frac{dp(t)}{dt}| = \sqrt{(\frac{dx(t)}{dt})^2 + (\frac{dy(t)}{dt})^2}$.

The *minimal exposure path* problem is to find a path with the minimum exposure connecting two known locations I and F.

Definition 10.6 (Minimal exposure path) Among all the paths connecting I and F, the one with the minimal exposure value is called a minimal exposure path, denoted by P_E.

To an intruder, a minimum exposure path is the best stealthy path with the least expected detection probability. But for the defender, this is the worst-coverage, and it needs to take some measure to increase the path exposure, such as adding more nodes.

For some simple scenarios, Megerian and Koushanfar [17] provide analytical solution to find a P_E. For example, a single sensor is located within the center of a square sensor field, as shown in Fig. 10.4(a). The P_E between the two points $P(1, -1)$ and $Q(-1, 1)$ consists of (1) a straight line segment from P to $(1, 0)$, (2) a quarter circle arc from $(1, 0)$ to $(0, 1)$, and (3) another straight line segment

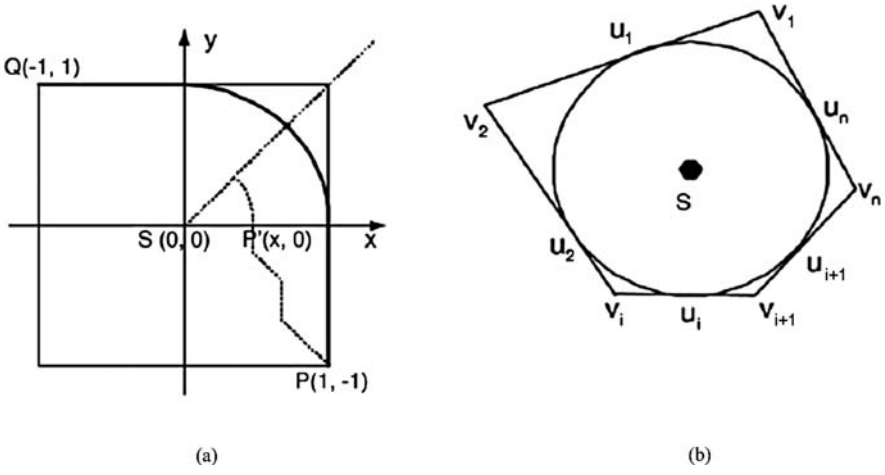

Fig. 10.4 (**a**) Minimal exposure path for a single sensor in a square sensor field; (**b**) minimal exposure path for a single sensor in a convex polygon sensor field (reproduced from [17], ©2002, Springer)

from (0, 1) to Q. Since any point on the dotted curve is closer to the sensor than any point lying on the straight line segments along the edges of the square, the exposure is more on the dotted curve. Also, since the length of the dotted curve is longer than the line segment, it would induce more exposure when an object travels along it, given that the time duration is the same in both the cases. This method can be extended to the scenario where the sensor field is a convex polygon and the sensor is at the center of the inscribed circle, as illustrated in Fig 10.4(b). Let the two curves between points v_i and v_j of the polygon be described as

$$\Gamma_{ij} = \overline{v_i u_i} * \overset{\frown}{u_i u_{i+1}} * \overset{\frown}{u_{i+1} u_{i+2}} * \cdots * \overset{\frown}{u_{j-2} u_{j-1}} * \overline{u_{j-1} v_j},$$

$$\Gamma'_{ij} = \overline{v_i u_{i-1}} * \overset{\frown}{u_{i-1} u_{i-2}} * \overset{\frown}{u_{i-2} u_{i-3}} * \cdots * \overset{\frown}{u_{j+1} u_j} * \overline{u_j v_j},$$

where $\overline{v_i u_i}$ is the straight line segment from point v_i to u_i, $\overset{\frown}{u_i u_{i+1}}$ is the arc on the inscribed circle between the two consecutive points u_i and u_{i+1}, and $*$ denotes a concatenation. The minimal exposure path between v_i and v_j is either Γ_{ij} or Γ'_{ij} whichever has less exposure.

However, finding a P_E in a sensor network with arbitrary deployment is an extremely difficult optimization task. Megerian et al. [17] propose to use a grid to transform the problem from the continuous domain to a tractable discrete domain. The weight of a line segment is computed as the exposure of this line segment by assuming a constant moving speed. The search for P_E is restricted to only the line segments of the grid and is found as the shortest path connecting I and F on this grid graph.

10.4 Detection Path

The coverage measure used for computing path breach and path support is defined as the Euclidean distance of a point to its closest sensor, which implicitly assumes that a point is sensed or covered by only one sensor (the best one in terms of the closest one). The coverage measure used for calculating path exposure allows that a point can be sensed by more than one sensor, which considers a simple form of collaborative sensing and processing capability among sensors. For some applications with known signal processing paradigm, one may define the coverage measure as the outcome of collaborative signal processing among sensors.

Clouquer et al. [4] consider the following signal attenuation model and collaborative signal processing model to define coverage measure. Let θ denote the signal strength of an intruder, which is assumed as a constant. Suppose that an intruder is located at a point p. Let x_k denote the energy measurement of this intruder signal strength by sensor s_k, and it is given by

$$x_k = \frac{\theta}{[d(p, s_k)]^\alpha} + n_k, \tag{10.1}$$

where α is the signal attenuation exponent, and n_k denotes the additive measurement noise. The sensors collaborate to arrive at a consensus decision about whether an intruder is present, which can be done by two approaches, namely, *value fusion* and *decision fusion*.

In value fusion, one of the sensors gathers the measurements from the other sensors, adds them up and compares the sum to a threshold Θ. If the sum is larger than the threshold, it decides the existence of an intruder. Given that an intruder is at point p, the probability of a correct consensus decision about the existence of this intruder is given by

$$D_v(p) = \Pr\left[\sum_{k=1}^{K}\left(\frac{\theta}{[d(p,s_k)]^{\alpha}} + n_k\right) \geq \Theta\right].$$

$D_v(p)$ measures a kind of detection capability at point p by K sensors and can be used as the coverage measure.

In decision fusion, each individual sensor compares its energy measurement to a threshold Θ_1 to arrive at a local decision and then sends its local decision (1 for intruder present and 0 otherwise) to a sensor. This sensor adds-up these local decisions and compares the sum to another threshold Θ_2. If the sum is larger than the threshold Θ_2, it decides the existence of an intruder at point p. Let $h_k(p, s_k)$ denote the local decision by sensor s_k. Given that an intruder is present at point p, the probability of a correct decision (i.e., $h_k(p, s_k) = 1$) is given by

$$\Pr\big[h_k(p,s_k) = 1\big] = \Pr\left[\left(\frac{\theta}{[d(p,s_k)]^{\alpha}} + n_k\right) \geq \Theta_1\right].$$

Note that $\Pr[h_k(p, s_k) = 0] = 1 - \Pr[h_k(p, s_k) = 1]$. Given that an intruder is present at point p, the probability of a correct consensus decision about the existence of an intruder is given by

$$D_d(p) = \Pr\left[\sum_{k=1}^{K} h_k(p,s_k) \geq \Theta_2\right].$$

$D_d(p)$ provides another coverage measure for the detectable capability at point p.

Similar to other types of penetration path, the detectability of a penetration path P connecting an initial location I and a final location F can be defined as the net probability of detecting an intruder that moves along the path P. An intruder using a path P is not detected if and only if it is not detected at any time while it is on that path. Therefore, the net probability ($G(P)$) of not detecting a target moving along the path P is the product of the probabilities of no detection at each point $p \in P$. In the logarithm form, $G(P)$ is given by

$$\log G(P) = \sum_{p \in P} \log\big(1 - D(p)\big),$$

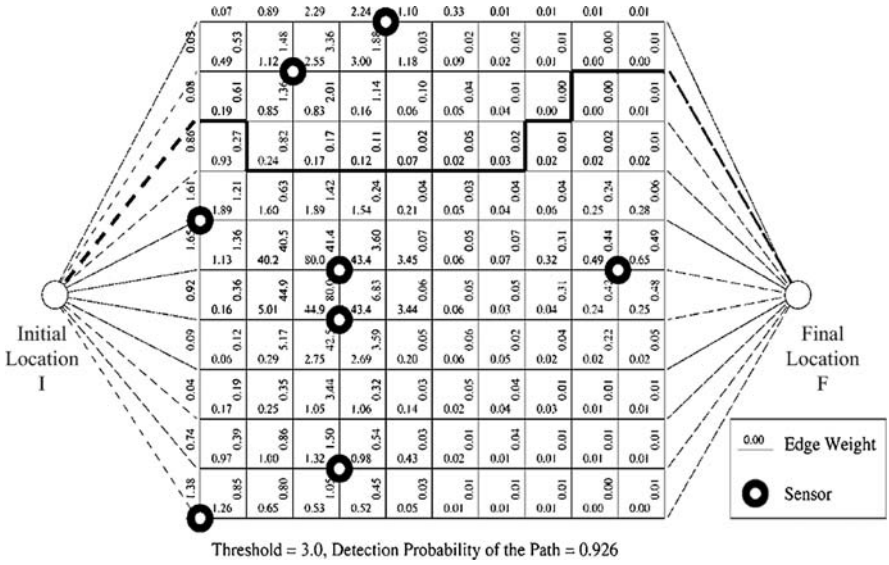

Fig. 10.5 Illustration of using a grid approach and applying shortest-path algorithm to find a minimal detection path (*the bold line*) (reproduced from [4], ©2003, Springer)

where $D(p)$ is either $D_v(p)$ or $D_d(p)$. The *path detectability* hence is computed by $1 - G(P)$. Likewise, the problem of finding a *minimal detection path*, denoted by P_D, is to find a path which minimizes $(1 - G(P))$ (or equivalently minimizes $|\log G(P)|$).

In general, the path P_D can be of arbitrary shape. Similar to the approach to find a minimal exposure path, Clouquer et al. [4] apply a grid to restrict the search space, and a minimal detection path is only composed of the line segments of the grid. The weight of a line segment L is assigned as $w_L \doteq |\sum_{p \in L} \log(1 - D(p))|$ and some classical shortest-path algorithm is then applied to find a P_D connecting I and F. Figure 10.5, redrawn from [4], illustrates a grid and a P_D on the grid.

For a given grid, if an intruder moves with a constant velocity, then the weight of a line segment w_L is inversely proportional to the sensors' sampling rate. Since $\log(1 - D(p))$ is nonpositive, a larger sampling rate results in more points $p \in L$ included in the computation of w_L and hence leads to a greater value of w_L. Although using a higher sampling rate increases the path detectability, it also consumes more energy. To balance the energy consumption and the path detectability, Deng and Liu [6] propose a simple binary search method which sets a target path detectability and then applies bisearch to find a sampling rate as small as possible. When an intruder can move with variable velocities, the problem becomes more complicated, and a regular grid is not enough to reflect the detectability of a line segment. Clouquer et al. [3] propose to extend a regular grid by adding more edges between two grid vertices, and the weights of these new edges reflect the detectability under different speeds.

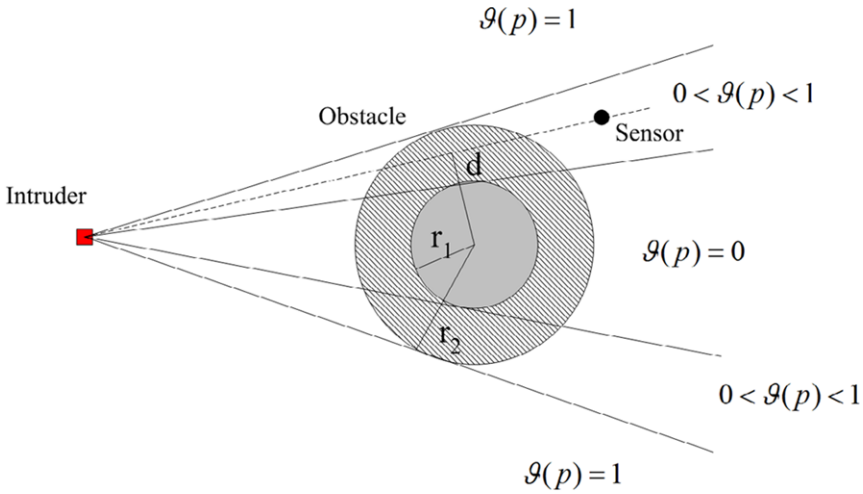

Fig. 10.6 Illustration of the distortion factor

Some researchers have proposed to use other collaborative signal processing paradigms to define coverage measure [2, 22, 23, 26]. For example, Onur et al. [23] consider using Neyman–Pearson detector (NP-detector) for collaborative detection. The NP-detector maximizes the detection probability while guaranteeing the maximal false alarm rate less than a required threshold. Wang et al. [26] define a coverage measure from the viewpoint of collaborative signal estimation. For the measurement model (10.1), the signal strength can be estimated from K sensors' measurements. If the estimation error is small, then the information about an intruder can also be obtained within some confidence level. Since the estimation error is actually a random variable due to the random measurement noise, Wang et al. [26] define the probability that the absolute value of estimation error is less than some threshold as the coverage measure for a point.

Chin et al. [2] propose a simple model to take into account the impact of obstacles. The signal measurement is distorted if there are obstacles in between the sensor and the intruder

$$x_k = \vartheta_k(p) \times \frac{\theta}{[d(p, s_k)]^\alpha} + n_k,$$

where $\vartheta_k(p)$ is the *distortion factor*. Each obstacle is modeled to be circular in shape and characterized by two radii r_1 and r_2. As illustrated by Fig. 10.6, a node which lies in the inner cone beyond the obstacle has distortion factor equal to zero. A node which lies outside the outer cone has distortion factor equal to one. Between the two cones, $\vartheta_k(p)$ increases linearly from zero to one with the distance $d - r_1$, where d is the distance from the center of the obstacle to the line joining the target and the node. When there are multiple obstacles between the target and the node, the distortion factor is assumed to be the product of the distortion effect of each obstacle.

10.5 Analysis for Path Characteristics

The objective of finding a penetration path is to find a route, given the initial location and final location of the route, within the sensor field such that the coverage measure of the path satisfies a predefined requirement. In general, the penetration paths, such as the maximal breach and support path, and exposure and detection path discussed in previous sections, are with curvilinear shapes. In addition to finding a qualified penetration path, a complementary study of path coverage is to analyze the properties of any given path in a sensor network. For example, given a path, what is the fraction it is covered? what is the probability of the whole path being completely covered? In the literature, the analysis for path characteristics have been carried out for straight line paths in a randomly deployed network [11, 16, 24].

Ram et al. [24] prove that the coverage process on any straight line path in a two-dimensional Boolean field is a one-dimensional Boolean process or, equivalently, a $M/G/\infty$ queue. A two-dimensional Boolean field is used to model a sensor field where sensors are deployed according to a spatial Poisson process of density λ and each sensor uses a Boolean disk coverage model. An n-dimensional Boolean process is a collection of random sets $X_i + C_i$, where $X_i \in \mathcal{R}^n$ are the locations of a Poisson point process, C_i are identically distributed, independent subsets of $\mathcal{R}^n$, and $X_i + C_i \equiv \{X_i + x : x \in C_i\}$ [10].

Let X_i denote the location of a sensor, and let R_i denote its sensing range. $\{R_i\}$ is assumed to be a sequence of positive independent identically distributed (i.i.d.) random variables with the support in $[0, 1]$. Let $f_{R_i}(r)$ denote the density of R_i, and let $\beta \doteq \mathbb{E}[R_i]$. For any given straight line, its coverage process consists of two components, namely, clumps and gaps, as illustrated by Fig. 10.7. For a sensor whose sensing disk intersects with the line, let $\overline{X}_i$ denote its projected location on the line, and let $\overline{R}_i$ denote its projected sensing radius on the line. Ram et al. [24] prove that the resulting $\{\overline{X}_i\}$ is a Poisson point process on the line with rate $\overline{\lambda} = 2\lambda\beta$. Since R_i is independent on X_i, $\overline{R}_i$ is also independent on $\overline{X}_i$. The following theorem states that the clumps of a line is a one-dimensional Boolean process.

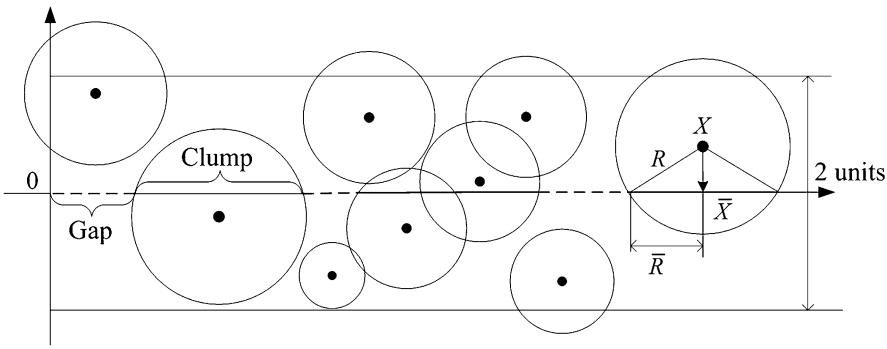

Fig. 10.7 Illustration of the coverage process on a straight line in a two-dimensional Boolean field

Theorem 10.3 (Ram et al. [24], Theorem 1) *The projected point process* $\overline{X}_i$ *and the collection of sensed segments form a one-dimensional Boolean process with laws identical to the one-dimensional Boolean process* $\{\overline{X}_i + \overline{C}_i\}$, *where* $\{\overline{X}_i\}$ *is a Poisson point process of density* $\overline{\lambda} = 2\lambda\beta$, $\overline{C}_i$ *is the random interval* $[0, 2\overline{R}_i]$, *and* $\{\overline{R}_i\}$ *are i.i.d. random variables with density*

$$f_{\overline{R}}(\overline{r}) = \begin{cases} \frac{\overline{r}}{\beta} \int_{\overline{r}}^{1} \frac{f_R(r)}{\sqrt{r^2 - \overline{r}^2}} dr & \text{for } 0 \le \overline{r} \le 1, \\ 0 & \text{otherwise.} \end{cases}$$

From the $M/G/\infty$ analogy, this theorem also says that the coverage process on a line is statistically equivalent to an $M/G/\infty$ queue with arrival rate $\overline{\lambda} = 2\lambda\beta$ and service time density given by

$$g(x) = \begin{cases} \frac{x}{4\beta} \int_{\frac{x}{2}}^{1} \frac{f_R(r)}{\sqrt{r^2 - \frac{x^2}{4}}} dr & \text{for } 0 \le x \le 2, \\ 0 & \text{otherwise.} \end{cases}$$

The event that a fraction ρ of the line segment is sensed by exactly k sensors corresponds to he event that, in an $M/G/\infty$ queue, for a fraction ρ of an observation period of duration l_0, there are exactly k customers in the system. From the ergodicity of the $M/G/\infty$ queue, as $l_0 \to \infty$, this probability has a Poisson distribution with mean $\mu \doteq \overline{\lambda} 2\mathbb{E}[\overline{R}]$. Therefore, for a straight line, the limiting fraction of the line that will be k-sensed is $\sum_{i=k}^{\infty} \frac{\mu^i e^{-\mu}}{i!}$. Some other properties of the path coverage can also be derived. For example, the critical density to guarantee that at least ρ fraction of every line should be k-sensed.

References

1. Adriaens, J., Megerian, S., Potkonjak, M.: Optimal worst-case coverage of directional field-of-view sensor networks. In: IEEE 3rd Annual Communications Society on Sensor and Ad Hoc Communications and Networks (SECON), pp. 336–345 (2006)
2. Chin, T.L., Ramanathan, P., Saluja, K.K., Wang, K.C.: Exposure for collaborative detection using mobile sensor networks. In: IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS), pp. 1–8 (2005)
3. Clouqueur, T., Ramanathan, P., Saluja, K.K.: Exposure of variable speed targets through a sensor field. In: IEEE International Conference on Information Fusion (IF), pp. 599–605 (2003)
4. Clouqueur, T., Phipatanasuphorn, V., Ramanathan, P., Saluja, K.K.: Sensor deployment strategy for detection of targets traversing a region. Mobile Networks and Applications **8**(4), 453–461 (2003)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd Edition. MIT Press, Cambridge (2001)
6. Deng, K., Liu, Z.: Minimum path exposure and detection interval setting for target detection using wireless sensor network. International Journal of Wireless Information Networks (Springer) **14**(4), 289–294 (2007)
7. Duttagupta, A., Bishnu, A., Sengupta, I.: Optimization problems based on the maximal breach path measure for wireless sensor network coverage. In: The 3rd International Conference on Distributed Computing and Internet Technology (ICDCIT), also in LNCS, vol. 4317, pp. 27–40 (2006)

8. Duttagupta, A., Bishnu, A., Sengupta, I.: Maximal breach in wireless sensor networks: Geometric characterization and algorithms. In: The 3rd International Workshop on Algorithmic Aspects of Wireless Sensor Networks (Algosensors), also in LNCS, vol. 4837, pp. 126–137 (2008)
9. Fang, C., Low, C.P.: Redundant coverage in wireless sensor networks. In: IEEE International Conference on Communications, pp. 3535–3540 (2007)
10. Hall, P.: Introduction to the Theory of Coverage Processes. Wiley, New York (1988)
11. Harada, J., Shioda, S., Saito, H.: Path coverage properties of randomly deployed sensors with finite data-transmission ranges. Computer Networks (Elsevier) **53**(7), 1014–1026 (2009)
12. Hou, Y.T., Chen, C.M., Jeng, B.: An optimal new-node placement to enhance the coverage of wireless sensor networks. Wireless Networks (Springer) (2009). doi:10.1007/s11276-009-0186-x
13. Huang, H., Richa, A.W., Segal, M.: Dynamic coverage in ad-hoc sensor networks. Mobile Networks and Applications **10**, 9–18 (2005)
14. Huang, L.S., Xu, H.L., Wang, Y., Wu, J.M., Li, H.: Coverage and exposure paths in wireless sensor networks. Journal of Computer Science and Technology **21**(4), 490–495 (2006)
15. Li, X.Y., Wan, P.J., Frieder, O.: Coverage in wireless ad hoc sensor networks. IEEE Transactions on Computers **52**(6), 753–763 (2003)
16. Manohar, P., Ram, S.S., Manjunath, D.: Path coverage by a sensor field: The nonhomogeneous case. ACM Transactions on Sensor Network (ToSN) **5**(2), 1–26 (2009)
17. Megerian, S., Koushanfar, F.: Exposure in wireless sensor networks: Theory and practical solutions. Wireless Networks **8**, 443–454 (2002)
18. Megerian, S., Koushanfar, F., Potkonjak, M., Srivastava, M.B.: Worst and best-case coverage in sensor networks. IEEE Transactions on Mobile Computing **4**(1), 84–92 (2005)
19. Meguerdichian, S., Koushanfar, F., Potkonjak, M., Srivastava, M.B.: Coverage problems in wireless ad-hoc sensor networks. In: IEEE Infocom, vol. 3, pp. 1380–1387 (2001)
20. Meguerdichian, S., Koushanfar, F., Qu, G., Potkonjak, M.: Exposure in wireless ad hoc sensor networks. In: ACM International Conference on Mobile Computing and Networking (MobiCom), pp. 139–150 (2001)
21. Mehta, D.P., Lopez, M.A., Lin, L.: Optimal coverage paths in ad-hoc sensor networks. In: IEEE International Conference on Communications (ICC), pp. 507–511 (2003)
22. Onur, E., Ersoy, C., Delic, H.: Finding sensing coverage and breach paths in surveillance wireless sensor networks. In: IEEE Internal Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 984–988 (2004)
23. Onur, E., Ersoy, C., Delic, H.: How many sensors for an acceptable breach detection probability? Computer Communications (Elsevier) **29**(2), 173–182 (2006)
24. Ram, S.S., Manjunath, D., Iyer, S.K., Yogeshwaran, D.: On the path coverage properties of random sensor networks. IEEE Transactions on Mobile Computing **6**(5), 446–458 (2007)
25. Veltri, G., Huang, Q., Qu, G., Potkonjak, M.: Minimal and maximal exposure path algorithms for wireless embedded sensor networks. In: ACM International Conference on Embedded Networked Sensor Systems (SenSys), pp. 40–50 (2003)
26. Wang, B., Chua, K.C., Wang, W., Srinivasan, V.: Worst and best information exposure paths in wireless sensor networks. In: International Conference on Mobile Ad-hoc and Sensor Networks (MSN05), also in LNCS, vol. 3794, pp. 52–62 (2005)

Appendix A
Voronoi Diagram and Delaunay Triangulation

A.1 Voronoi Diagram

The Voronoi diagram is a versatile geometric structure and has many applications in social geography, physics, astronomy, robotics, and many more fields [1, 2]. Let $\mathcal{S}$ denote a set of n *sites* (e.g., sensor nodes) in the two-dimensional plane. For two distinct sites $p, q \in \mathcal{S}$, the *dominance* of p over q is defined as the subset of the plane being at least as close to p as to q, that is,

$$\mathrm{dom}(p, q) = \left\{ x \in \mathbb{R}^2 \mid d(x, p) \leq d(x, q) \right\}$$

where $d(x, p)$ is the Euclidean distance between two points x and p. Let $\overline{pq}$ denote the line segment between p and q. The perpendicular bisector of $\overline{pq}$ divides the plane into two halves. This perpendicular bisector, denoted by $\overline{pq}^{\perp}$, is called the *separator* of p and q. We denote the open half-plane that contains p by $h_p(q)$. Any point on the separator $\overline{pq}^{\perp}$ is equidistant to p and q. Any point within a half plane $h_p(q)$ has the distance to p smaller than the distance to q. Therefore, $\mathrm{dom}(p, q)$ is the perpendicular bisector line $\overline{pq}^{\perp}$ plus the half of the plane $h_p(q)$. The *region* of a site $p \in \mathcal{S}$ is the portion of the plane lying in all the dominance of p over the remaining sites in $\mathcal{S}$, that is,

$$\mathrm{reg}(p) = \bigcap_{q \in \mathcal{S} - \{p\}} \mathrm{dom}(p, q)$$

The region $\mathrm{reg}(p)$ comes from intersecting $n - 1$ half planes, and it is a convex polygon. Furthermore, the boundary of $\mathrm{reg}(p)$ consists of at most $n - 1$ *edges* (maximal open straight-line segments) and *vertices* (their endpoints). Each point on an edge is equidistant from exactly two sites, and each vertex is equidistant from at least three sites. As a consequence, the edges and vertices of all regions form a polygonal division of the whole plane. This partition is call the *Voronoi diagram*, $\mathrm{Vor}(\mathcal{S})$. These edges are called *Voronoi edges* and the vertices are called *Voronoi*

B. Wang, *Coverage Control in Sensor Networks,*
Computer Communications and Networks,
DOI 10.1007/978-1-84996-059-5, © Springer-Verlag London Limited 2010

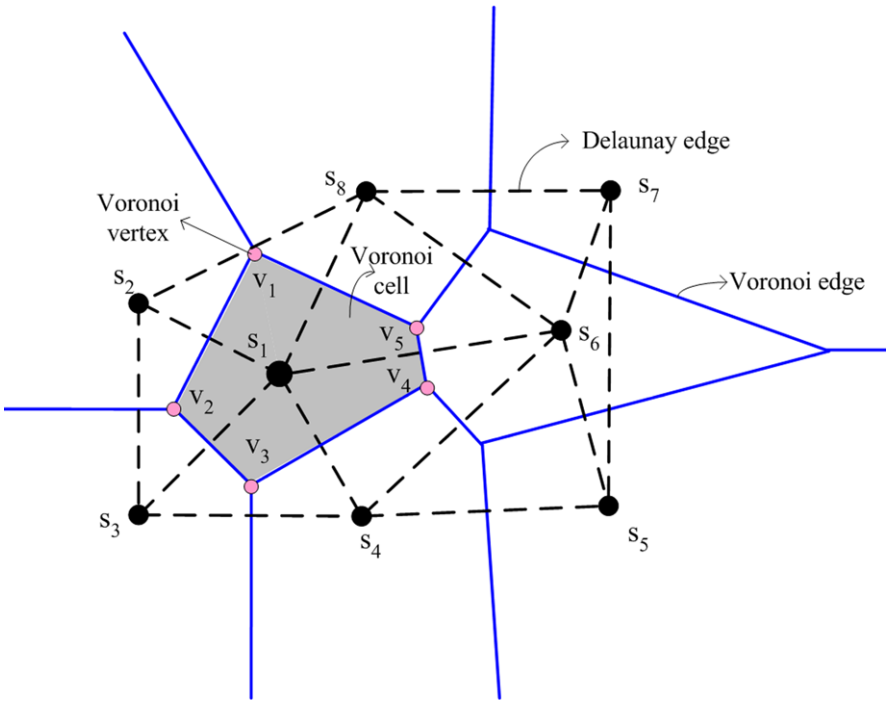

Fig. A.1 Illustration of Voronoi diagram and Delaunay triangulation for 8 sites

vertices. A region $\mathrm{reg}(p)$ is called a *Voronoi cell* (or *Voronoi polygon*). Fig. A.1 illustrates these concepts.

For n sites, the Voronoi diagram $\mathrm{Vor}(\mathcal{S})$ contains exactly n Voronoi cells. Some of them are necessarily unbounded. There are no Voronoi vertices if and only if all sites in $\mathcal{S}$ lie on a single straight line. Although n sites give rise to $\binom{n}{2} = O(n^2)$ separators, not all separators define edges of $\mathrm{Vor}(\mathcal{S})$ and not all intersections are vertices of $\mathrm{Vor}(\mathcal{S})$. In fact, it can be shown that for $n \geq 3$, the number of vertices in the Voronoi diagram of a set of n sites in the plane is at most $2n - 5$, and the number of edges is at most $3n - 6$ [2].

A straightforward way to construct the Voronoi diagram is to do this: for each site p, compute the common intersection of the half-planes $h_p(q_i), q_i \in \mathcal{S}, q_i \neq p$. This way we spend $O(n \log n)$ time per Voronoi cell, leading to an $O(n^2 \log n)$ algorithm to compute the whole Voronoi diagram. Another commonly known *Fortune's algorithm*, which uses a sweep line to scan all sites from top to bottom of the plane, computes the Voronoi diagram in $O(n \log n)$ time. It can be shown that the problem of sorting n real numbers is reducible to the problem of computing Voronoi diagram. Therefore, any algorithm for computing Voronoi diagrams must take $\Omega(n \log n)$ time in the worst case and Fortune's algorithm is an optimal one in this regard [2].

A.2 Delaunay Triangulation

Let $\mathcal{S} \doteq \{s_1, s_2, \ldots, s_n\}$ be a set of sites (points) in the plane. We can connect any two sites with a straight line segment to form a subdivision of the plane. A *maximal planar subdivision* is defined as a subdivision $\mathcal{D}$ of the plane such that no edge connecting two vertices can be added to $\mathcal{D}$ without destroying its planarity. In other words, any edge that is not in $\mathcal{D}$ intersects one of the existing edges. A *triangulation* of $\mathcal{S}$ is defined as a maximal planar subdivision whose vertex set is $\mathcal{S}$. The Delaunay triangulation is one of triangulations of $\mathcal{S}$ with some useful properties and is closely related to the Voronoi diagram.

The planar Delaunay graph and the planar Voronoi diagram are duals in a graph theoretical sense. Voronoi vertices correspond to Delaunay triangles, and Voronoi cells correspond to Delaunay vertices. A Delaunay graph can be constructed as an embedding of the Voronoi diagram: connect two sites in the plane with a straight line segment if and only if their Voronoi cells share a common edge. As illustrated by Fig. A.1, the Delaunay edges (dashed) are orthogonal to their corresponding Voronoi edges (solid), but do not necessarily intersect them. Furthermore, the boundary of the convex hull of the sites consists of Delaunay edges.

The Delaunay graph is a planar graph, i.e., no two Delaunay edges in the graph cross. If the set of sites $\mathcal{S}$ contains no four points on a circle, then $\mathcal{S}$ is called in *general position*. If $\mathcal{S}$ is in general position, all vertices of the Voronoi diagram have degree three, and consequently all bounded faces of the Delaunay graph are triangles and the Delaunay graph is called Delaunay triangulation of $\mathcal{S}$. Furthermore, if $\mathcal{S}$ is in general position, the Delaunay triangulation is unique and maximizes the minimum internal angle over all triangulations of $\mathcal{S}$. Besides using the Voronoi diagram, the Delaunay triangulation can also be computed directly in $O(n \log n)$ expected time [2].

References

1. Aurenhammer, F.: Voronoi diagrams—A survey of a fundamental geometric data structure. ACM Computing Surveys **23**(4), 345–406 (1991)
2. Berg, M., Cheong, O., Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications. Springer, Berlin (2008)

Index

B. Wang, *Coverage Control in Sensor Networks,*
Computer Communications and Networks,
DOI 10.1007/978-1-84996-059-5, © Springer-Verlag London Limited 2010

Color Plates

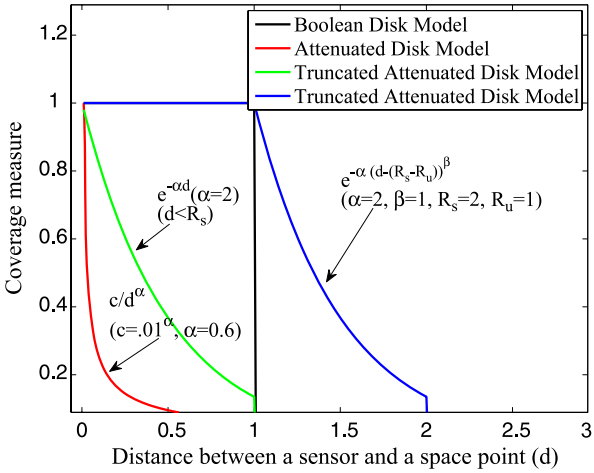

Fig. 2.5 Illustration of the relation between the coverage measure and sensor–point distance for (**a**) Boolean disk coverage model; (**b**) attenuated disk coverage model; and (**c**) and (**d**) truncated attenuated disk coverage model

B. Wang, *Coverage Control in Sensor Networks,*
Computer Communications and Networks,
DOI 10.1007/978-1-84996-059-5, © Springer-Verlag London Limited 2010

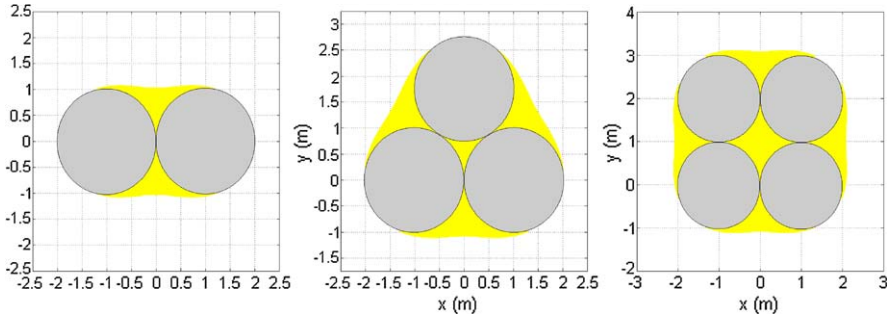

Fig. 2.6 Illustration of the space points covered by using the detection coverage model of (**a**) 2 sensors, (**b**) 3 sensors, and (**c**) 4 sensors

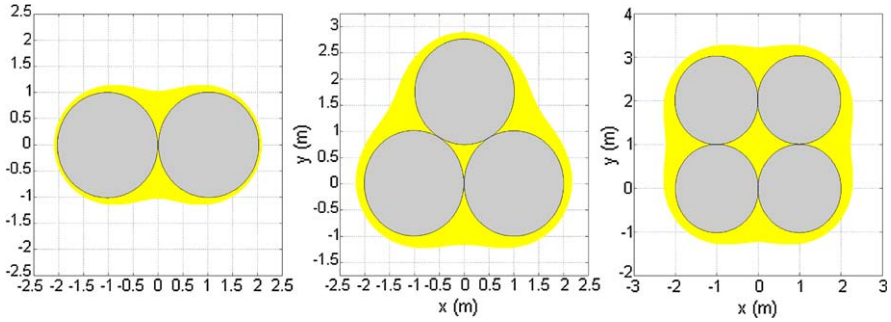

Fig. 2.7 Illustration of the space points covered by using the estimation coverage model of (**a**) 2 sensors, (**b**) 3 sensors, and (**c**) 4 sensors

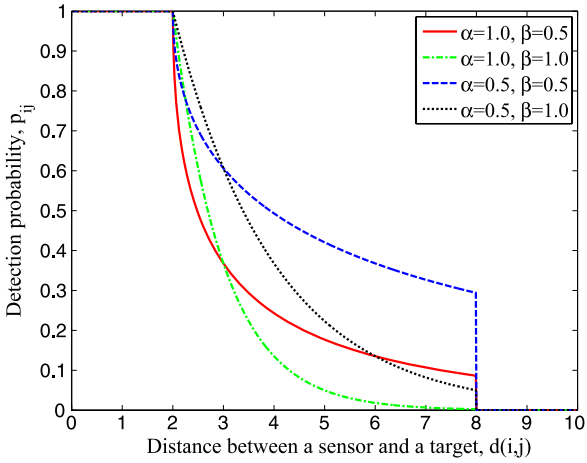

Fig. 4.4 Illustration of the detection model with two cut-off thresholds

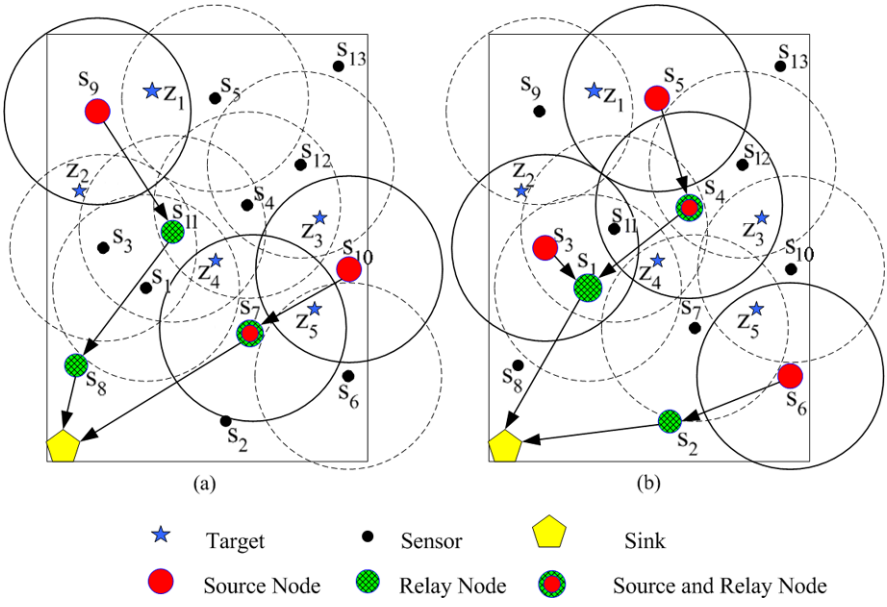

Fig. 5.4 Illustration of the connected target coverage problem: (**a**) one potential solution; (**b**) another potential solution

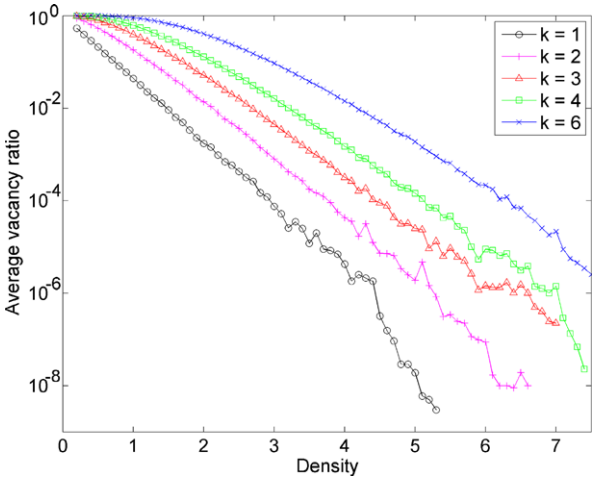

Fig. 6.8 Average k-vacancy ratio vs. sensor density

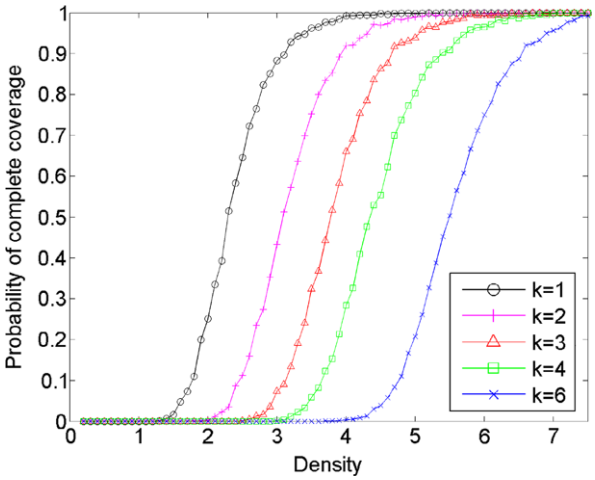

Fig. 6.9 Probability of complete k-coverage vs. sensor density