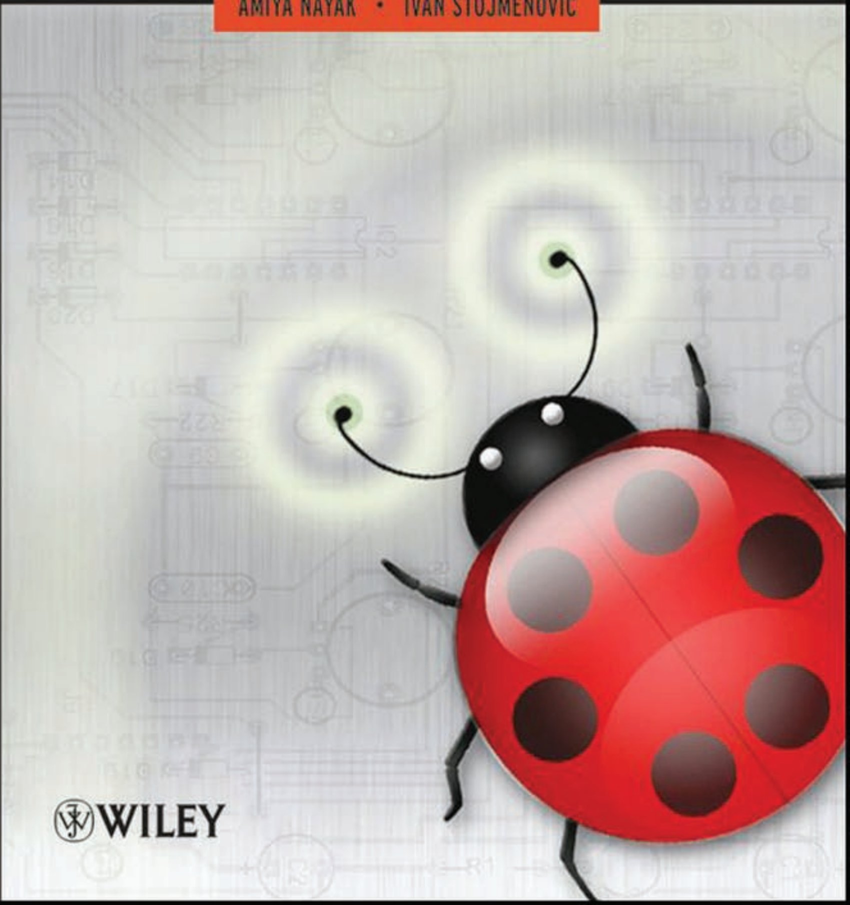
Wireless Sensor and Actuator Networks

Algorithms and Protocols for Scalable Coordination and Data Communication

Amiya Nayak • Ivan Stojmenovic



WILEY

Wireless Sensor and Actuator Networks

Wireless Sensor and Actuator Networks

Algorithms and Protocols for Scalable Coordination and Data Communication

Edited by

Amiya Nayak and Ivan Stojmenovic

WILEY

A JOHN WILEY & SONS, INC., PUBLICATION

***Library of Congress Cataloging-in-Publication Data*:**

Wireless sensor and actuator networks : algorithms and protocols for scalable coordination and data communication / edited by Amiya Nayak, Ivan Stojmenovic.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-470-17082-3 (cloth)
1. Wireless sensor networks. 2. Computer algorithms. 3. Computer network protocols. I. Nayak, Amiya. II. Stojmenovic, Ivan.
TK7872.D48W5666 2010
681′.2–dc22
2009031421

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Cover page designed by Milos Stojmenovic.

Contents

Preface

Traditional and existing sensor and actuator networks use wired communications, whereas, wireless sensors provide radically new communication and networking paradigms, and myriad new applications. They have small size, low battery capacity, non-renewable power supply, small processing power, limited buffer capacity (thus routing tables, if used at all, must be small), a low-power radio, and lack unique identifiers. Sensors may measure distance, direction, speed, humidity, wind speed, soil makeup, temperature, chemicals, light, vibrations, motion, seismic data, acoustic data, strain, torque, load, pressure, and so on. These nodes are autonomous devices with integrated sensing, processing, and communication capabilities. Nodes in a sensor network are generally densely deployed. Thousands of sensors may be placed, mostly at random, either very close or inside a phenomenon to be studied. Once deployed, the sensors are expected to self-configure into an operational wireless network, and must work unattended. The limited energy budget at the individual sensor level implies that in order to ensure longevity, the transmission range of individual sensors needs to be restricted. In turn, this implies that wireless sensor networks must be multi-hop ones.

Current research and implementation efforts are mostly oriented toward a traditional scenario with stationary sensors and a single static sink that collects information from sensors where the sink is directly connected to the user (or task manager). However, the latest research has unearthed the practically unsolvable problem of uneven energy distribution and energy holes in this scenario. Therefore, generalized scenarios have been considered, such as sensor networks with multiple stationary sinks, single mobile sink, or multiple mobile sinks. Mobile sensors have also been discussed. And, we envision adding actuators to the network. The difference between sinks and actuators is that actuators are able to act on the environment; mobile actuators may additionally act on the sensors. Actuators may also perform the roles of sinks, or both sinks and actuators may coexist in a given network. We will now elaborate on aspects of actuation.

Sensor–actuator networks are heterogeneous networks that comprise networked sensor and actuator nodes which communicate among each other using wireless links to perform distributed sensing and actuation tasks. *Actuators* (called also *actors*) are resource-rich, potentially mobile, and are involved in taking decisions and performing appropriate actions on themselves (e.g. controlled movement), on sensors (such as activating sensors, moving or replacing a sensor), and/or in the environment (e.g. turn on their own a water sprinkler to stop the fire). Sensor–actuator networks are expected to operate autonomously in unattended environments. They may be directly connected (using, for instance, web infrastructure) and responsive to a user (task manager) who controls the network via sinks. One or more actuator(s) may also play the role of sink(s). In fact, sinks can be treated as special kinds of actuators, although a better interpretation might be to associate them with base stations that communicate directly with the user.

Since the actuating task is a more complicated and energy-consuming activity than the sensing task, actuators are resource-rich nodes equipped with better processing capabilities, higher transmission powers, and longer battery life. Moreover, depending on the application, there may be a need to rapidly respond to sensor input. Therefore, the issue of real-time communication is very important since actions are performed on the environment after the sensing occurs. In addition, while the number of sensor nodes deployed to achieve a specific application objective may be in the order of hundreds or thousands, such a dense deployment is not necessary for actuator nodes due to the different coverage requirements and physical interaction methods of acting a task. Hence, the number of actuators is much less than that of sensors.

The goal of this book is to present a fault-tolerant, reliable, low latency, and energy-aware framework for wireless sensor and actuator networks, so that the ultimate goal of their applications (protecting critical infrastructures, enabling timely emergency responses, and environment monitoring) can be fulfilled. Future sensor–actuator networks will be more heterogeneous and radically distributed, potentially with millions of nodes. They may respond to multiple tasks, to multiple and potentially mobile sinks and/or actuators, and multiple sensor networks may be integrated into a single network. There are algorithmic challenges in the rapidly emerging field of future heterogeneous super-networks where sensor networks will be integrated into wired and/or wireless infrastructure. Challenges of such wide-area sensor systems include scalability, robustness, manageability, and actuation. Having this futuristic vision in mind, this book will provide a protocol framework at the network layer, namely data communication and coordination issues. While being general, the framework should generate optimal solutions when applied/mapped in any specific emerging application; that is, the very same protocols may be applied in scenarios ranging from a simple scenario with one fixed actuator to the envisioned super-networks. To achieve such ambitious goals, several primary criteria for protocol design must be followed: energy consumption, localized design, reliability, parameterless behavior, and simplicity.

This book is problem-oriented, with each chapter discussing computing and communication problems and solutions that arise in rapidly emerging wireless sensor and actuator networks. The main direction of the book is to review various algorithms and protocols that have been developed in the area with emphasis on the most recent ones. The book is intended to cover a wide range of recognized problems in sensor–actuator networks, striking a balance between theoretical and practical coverage. The theoretical contributions are limited to the scenarios and solutions that are believed to have practical relevance. This book is unique in addressing sensor and actuator/actor networking in a comprehensive manner, covering all the aspects, and providing up-to-date information. It is an appropriate and timely forum, where industry, operators, and academics from several different areas can learn more about current trends and become aware of the possible architectures of sensor and actuator networks, their advantages, and their limits in future commercial, social, and educational applications.

This book is intended for researchers and graduate students in computer science and electrical engineering, and researchers and developers in the telecommunication industry. It is directed at those who are looking for a reference resource in sensor and actuator networking and those who want to get a global view of this area.

The book is based on a number of stand-alone chapters that together cover the subject matter in a fully comprehensive manner. As a result of the exponential growth in the number of studies, publications, conferences, and journals on sensor networks, a number of graduate courses fully or partially concentrating on sensor networks have emerged recently. It is expected that this book will act as a supplemental textbook for such graduate courses. It can be also used as a stand-alone textbook for a course specifically on wireless sensor and actuator networks. The chapters cover subjects describing state-of-the-art approaches and surveying the existing important solutions. They provide readable but informative content, with appropriate illustrations, figures, and examples. A number of chapters also provide some problems and exercises for use in graduate courses.

The book content addresses the dynamic nature of wireless sensor and actuator networks. Due to frequent node addition and deletion from networks (changes between active and sleeping periods, done to conserve energy, are one of the contributors to this dynamic), and possible node movement, the algorithms that can be potentially used in real equipments must be *localized* and must have minimal communication overhead. The overhead should consider both the construction and its maintenance for the structure used in solutions and ongoing protocols. We believe that this is the only approach that will eventually lead to the design of protocols for real applications. We will explain now our design principles and the priorities given to the coverage of topics in this book.

A *scalable* solution is one that performs well in a large network. Sensor networks may have hundreds or thousands of nodes. Priority is given to protocols that perform well for small networks, and perform significantly better for large networks (more precisely, are still working as opposed to crashing when other methods are applied). In order to achieve scalability, new design paradigms must be applied. The main paradigm shift is to apply localized schemes as opposed to most existing protocols that require global information. In a *localized* algorithm, each node makes protocol decisions solely based on knowledge about its local neighbors. In addition, the goal is to provide protocols that will minimize the number of messages between nodes, because bandwidth and power are limited. Protocols should use a small constant number of messages, often even none beyond preprocessing 'hello' messages. Localized message-limited protocols provide scalable solutions. Typical local information to be considered is one-hop or two-hop neighborhood information (information about direct neighbors and possibly the neighbors of neighbors). Nonlocalized distributed algorithms, on the other hand, typically require global network knowledge, including information about the existence of every edge in the graph. The maintenance of global network information, in the presence of mobility or changes between sleep and active periods, imposes a huge communication overhead, which is not affordable for bandwidth- and power-limited nodes. In addition to being localized, protocols are also required to be *simple, easy to understand and implement*, and to have *good average case performance*. Efficient solutions often require position information. It was widely recognized that sensor networks can function properly only if reasonably accurate position information is provided to the nodes.

BRIEF OUTLINE OF THIS BOOK

This book consists of 10 chapters. It begins with an introductory chapter that describes various scenarios where sensor and actuator networks may be applied, problems at physical, medium access, network, and transport layers, and various application layer tools for enabling applications. It argues for the use of localized algorithms, and discusses the generation of sensor and actuator networks for simulation purposes.

Chapter 2 discusses backbones as subsets of sensors or actuators that suffice for performing basic data communication operations. They are applied for energy-efficient data dissemination tasks. The goal is to minimize the number of re-broadcasts while attempting to deliver messages to all sensors or actuators. Neighbor detection and route discovery algorithms that consider a realistic physical layer are described. An adaptive broadcasting protocol without parameters, suitable for delay-tolerant networks, is further discussed. We also survey existing solutions for the minimal energy broadcasting problem where nodes can adjust their transmission powers.

Sensor networks normally have redundancy for sensing coverage. Some sensors are allowed to sleep while preserving network functionality. Sensors should decide which of them should be active and monitor an area, and which of them may sleep and become active at a later time. Sensor area coverage problem has been considered for both the unit disk graph– and physical layer–based sensing models in Chapter 3. Actuators may similarly run a protocol to decide about their service areas, releasing some of them from their particular duty. Operational range assignment for both sensor and actuators nodes is also discussed.

Chapter 4 surveys existing flooding-based and position-based routing schemes. It also describes a general cost-to-progress ratio-based approach for designing routing protocols under a variety of metrics, such as hop count, power, remaining energy, delay, and others. Chapter 4 also describes routing with guaranteed delivery for unit disk graphs and ideal MAC layer based on the application of the Gabriel graph, a localized planar and connected structure. Solutions are expanded toward beaconless behavior, where nodes are not aware of their neighborhood. Georouting with virtual coordinates is based on hop distances to some landmarks. This chapter also discusses physical layer aspects of georouting, routing in sensor–actuator networks, and load balancing issues in routing.

Chapter 5 reviews the scenarios where a given message is sent from a single source (sensor) to possibly several destinations (actuators). These scenarios can be subdivided into multicasting, geocasting, multiratecasting, and anycasting. In multicasting, a given message must be routed from one node to a number of destinations whose locations may be arbitrary and spread over the network. Geocasting destinations are all nodes located in a given geographical area. Multiratecasting is a generalization of multicasting, where regular messages are sent from a source to several destinations, possibly at a different rate for each destination. Finally, in an anycasting scenario, a source must send a message to any node, preferably only one, among a given set of destinations. Each of these scenarios corresponds to a typical use case in sensor and actuator networks.

Data gathering aims to collect sensor readings from sensory fields at predefined sinks or actuators (without aggregating at intermediate nodes) for analysis and processing. Research has shown that sensors near a data sink deplete their battery power faster than those far apart, due to their heavy overhead of relaying messages. Nonuniform energy consumption causes degraded network performance and shortens network lifetime. Recently, sink mobility has been exploited to reduce and balance energy expenditure among sensors. The effectiveness has been demonstrated both by theoretical analysis and by experimental study. In Chapter 6, we investigate the theoretical aspects of the uneven energy depletion phenomenon around a sink/actuator, and address the problem of energy-efficient data gathering by mobile sinks/actuators. We present taxonomy and a comprehensive survey of the state of the art on the topic.

The efficiency of many sensor network algorithms depends on characteristics of the underlying connectivity, such as the length and density of links. The number and nature of links that are to be used among all potentially available links can be controlled. Topology control can be achieved by modifying the transmission radii, selecting a given subset of the links, or moving some nodes (if such functionality is available). Chapter 7 reviews some of these problems and related solutions, applicable to the context of sensor and actuator networks. Spanning structures and minimum weight connectivity are applied for power-efficient and delay-bounded data aggregation. Detection of critical nodes and links aims to provide fault tolerance to the applications. Some recent and prospective works considering biconnectivity of mobile sensors/actuators and related deployment of sensors, augmentation, area and point coverage are discussed.

In the location service problem, mobile actuators send location update messages, while stationary sensors send search messages to learn the latest position of actuators. The task is to minimize combined update and search message cost, while maximizing the success rate of finding a target actuator and subsequently routing to it. In the literature, many location service algorithms have been proposed for mobile *ad hoc* networks, and they can be directly applied to sensor and mobile actuator networks. Chapter 8 reviews research efforts on this topic.

Chapter 9 surveys the existing representative work in both, sensor–actuator and actuator–actuator coordination. Sensor–actuator coordination deals with establishing data paths between sensors and actuators, and can be used for sensor deployment. Actuator–actuator coordination includes robot coordination for sensor placement, dynamic task allocation, selecting the best robot to respond to reported events, robot dispersion, boundary coverage, and fault-tolerant response. In coordinated actuator movement problems, actuators are moved to desired locations to save energy in long-term communication tasks where the traffic is sufficiently regular and large in volume to warrant nodes expending energy for moving. Chapter 9 also reviews recent developments in coordination among flying robots.

Coverage is the functional basis of any sensor network. The impact on coverage from stochastic node dropping and inevitable node failure, coupled with controlled node mobility, gives rise to the problem of movement-assisted sensor placement in wireless sensor and actuator networks (WSAN). One or more actuators may carry sensors, and drop them at a proper position, while moving around, in the region of interest (ROI) to construct desired coverage. Mobile sensors may change their original placement so as to improve existing coverage. Emerging coverage holes are to be covered by idle sensors. Actuators may place spare sensors according to certain energy optimality criteria. If sensors are mobile, they can relocate themselves to fill holes. Chapter 10 comprehensively reviews existing solutions to the sensor placement problem in WSAN.

ACKNOWLEDGMENTS

The editors are grateful to all the authors for their valuable contribution to this book. The editors are thankful to Michael Christian from Wiley for his timely and professional cooperation, and for his decisive support of this project.

We hope that the readers will find this book informative and worth reading. Comments received by readers will be greatly appreciated.

Amiya Nayak and Ivan Stojmenović
SITE, University of Ottawa,
Ottawa, Ontario, Canada

Contributors

Arnaud Casteigts *School of Information Technology and Engineering, University of Ottawa, 800 King Edward, Ottawa, Ontario K1N 6N5, Canada*

Xu Li *School of Information Technology and Engineering, University of Ottawa, 800 King Edward, Ottawa, Ontario K1N 6N5, Canada*

Hai Liu *Hong Kong Baptist University, Hong Kong, P.R. China*

Veljko Malbasa *Faculty of Technical Sciences, University of Novi Sad, Serbia*

Ivan Mezei *Faculty of Technical Sciences, University of Novi Sad, Serbia*

Amiya Nayak *School of Information Technology and Engineering, University of Ottawa, 800 King Edward, Ottawa, Ontario K1N 6N5, Canada*

David Simplot-Ryl *CNRS/INRIA/University of Lille 1, France*

Ivan Stojmenovic *School of Information Technology and Engineering, University of Ottawa, 800 King Edward, Ottawa, Ontario K1N 6N5, Canada*

Chapter 1

Applications, Models, Problems, and Solution Strategies

Hai Liu[1], Amiya Nayak[2], and Ivan Stojmenovic[2]

[1]*Hong Kong Baptist University, Hong Kong, P.R. China*
[2]*School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5*

Abstract

This introductory chapter describes various applications, scenarios, and models of wireless sensor and actuator networks. Problems at the physical, medium access, network, and transport layers as well as various tools needed to enable their functioning are identified. Various assumptions and metrics used in simulations and protocol descriptions are discussed. The chapter then describes ways of generating sensor and actuator networks based on widely accepted unit disk graph models. Finally, this chapter discusses solution approaches arising in sensor networks and advocates the use of localized protocols, where individual sensors and actuators make their decisions based on local knowledge.

1.1 WIRELESS SENSORS

We will elaborate first on wireless sensors; then on wireless sensor networks (WSNs) (with a single sink), their properties, models, and application types. Afterwards, we will add actuators to the model and discuss various combinations of sensors and actuators that can form a heterogeneous network with different levels of complexity.

Recent technological advances have enabled the development of small-sized (a few cubic centimeters), low-cost, low-power, and multifunctional sensor devices. There are different types of sensors. Sensors are normally specialized, but sometimes a few capabilities may be available in a single sensor. They may measure distance, direction, speed, humidity, wind speed, soil makeup, temperature, chemical composition, light, vibration, motion, seismic activity, acoustic properties, strain, torque, load, pressure, and so on.

Traditionally, sensors are attached to the environment and their measurements are sent to a base station (BS) with wired communication. There exists a large body of knowledge on such models and applications of sensors which have been studied by a huge community of researchers. During the last decade, a new vision of sensor nodes as autonomous devices with integrated sensing, processing, and communication capabilities has emerged. Attaching antenna for receiving signals and a transmitter enables wireless communication of sensors. Sensors also have a small processor and a small memory for coding and decoding signals, as well as for running simple communication protocols. They differ in their battery capacity; for example, some of them run on small batteries and last a day, whereas others have larger batteries attached that let them last up to a month with continuous operation. In some applications, a renewable power supply such as a solar panel is used. Further, some sensors are embedded into other devices and draw their required energy from them. Such sensors do not have energy limitations in their functioning. In some scenarios, sensors could be provided with a wireless single-hop access to infrastructure networks such as the Internet.

For some applications, sensors may be of a large size, especially if they are protected by boxes or lifted to a height that improves their communication and protection level. When collected data is not time critical, sensors may function in isolation. For example, seismological data or bird presence detected acoustically can simply be collected in the local sensor memory and downloaded when visited by humans. This book concentrates, however, on scenarios involving networks of wireless sensors.

1.2 SINGLE-HOP WIRELESS SENSOR NETWORKS

The majority of the existing applications for “wireless” sensors rely on a single-hop wireless network to reach a BS for further processing of the measured phenomena. That is, sensor measurements are sent directly, using a wireless medium, from sensor to BS. Most of these applications rely on sensors that are *embedded* into a different device. Also, the majority of applications for embedded sensors rely on single-hop wireless communication. For example, small sensors can be embedded into a traffic surveillance system to monitor traffic on congested roads or be used to monitor hot spots in a region or building.

Health care is one of the primary applications for wireless networks composed of embedded sensors. Sensors can be embedded into watches which, when attached to patients, monitor and analyze data such as pulse and blood pressure. In case of potential health risks, individual sensors send alarm messages to

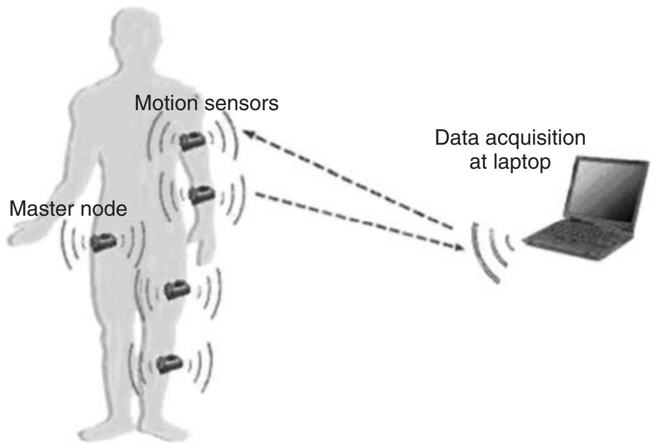


Figure 1.1 Monitoring limb movement in stroke patient rehabilitation.

a nearby control center via one-hop wireless communication. As these sensors are battery powered, they can benefit from intelligent sensor management that provides energy efficiency as well as quality of service (QoS) control.

For example, a wireless motion analysis sensor for stroke patient rehabilitation was studied in John *et al.* (2005). Wearable sensor motes with armbands were attached to stroke patients to monitor their limb movements and muscle activity during rehabilitation exercises. The sensor board consisted of a three-axis accelerometer, a gyroscope, and various electromyogram (EMG) sensors. It was able to capture a rich set of motion data used for studying the effects of various rehabilitation exercises on the patient population. The collected data was transmitted to a data acquisition or control center, such as a laptop or PC, via one-hop wireless communication. The system architecture is shown in Figure 1.1 below.

Some large-scale sensor networks may also be single-hop in terms of wireless communication needed for reporting. For example, the sink, or several sinks, could be mobile and move around the network. This would allow them to get close enough to the sensors so that report collecting could be done in a single hop. In other examples, embedded sensors could move toward a fixed sink. For example, sensors can be embedded into sea mammals to trace their locations over time. When a sea mammal approaches a fixed BS, reports can be downloaded.

1.3 MULTIHOP WIRELESS SENSOR NETWORKS

Nodes in the WSNs are generally randomly and densely deployed. For example, thousands of sensor nodes may be dropped from airplanes to monitor an interested area. Once deployed, these sensors are expected to self-configure into a wireless network. Since the energy budget of an individual sensor is very limited, the transmission range of sensors is also restricted. Thus, WSNs usually operate in a multihop fashion. A large number of sensor devices can be organized in a multihop fashion to provide unlimited potential to “sense” the physical world. Reports from individual sensors are sent to other sensors, where they can be combined with other sensor readings or simply retransmitted to other sensors until a sink node that is capable of communicating with a user is reached. Therefore, individual sensor readings may need several wireless hops to reach a BS. Such WSNs have received significant attention in recent years. A WSN usually consists of a large number of low-cost and low-energy sensor nodes, which can be deployed on the ground, in the air, in vehicles, or inside buildings. Nodes in WSNs sense data, find routes, and forward sensing data to a sink or BS that is usually far away from the data source. Since sensors usually have a small size, low-battery capacity, nonrenewable power supply, limited processing ability, small buffer capacity, and a low-powered radio, WSNs pose new challenges to both industrial and academic communities.

Applications for WSNs have been envisioned for a wide range of areas. These include, but are not limited to, the following: environment monitoring (e.g., traffic, habitat, security, etc.), infrastructure protection (e.g., power grids and water distribution), fire prevention, agriculture, health care, chemical plumes tracking, building monitoring or control, warehouse management, smart transportation, and context-aware computing (e.g., smart homes and responsive environments) as well as industrial sensing, diagnostics and process control, biomedical sensor engineering, water and waste management, military applications, and so on.

Most of the scenarios considered contain a single sink (also called *base station*), which is normally static. The sink in a WSN collects information from sensors and then analyzes and processes the information for specific applications. The sink could be connected to the Internet via wireless or wired communications such that a remote user is able to inquire about data via the Internet (at any time or from anywhere). Single sink scenarios, or scenarios with multiple fixed or mobile sinks, have also been explored in literature. Sensors in WSNs are usually static. However, they can be mobile when attached to robots, soldiers, or vehicles.

1.4 EVENT-DRIVEN, PERIODIC, AND ON-DEMAND REPORTING

There are three types of applications for WSNs and each has its corresponding data communication modes: *event-driven*, *periodic*, and *on-demand reporting*. In the *event-driven* mode, sensors report the sensing data to the sink once a specified event (e.g., fire) has been detected. In the *periodic reporting* (or *time-driven*) mode, sensor nodes gather information from the environment at predetermined times and periodically send the data to the sink. In the *on-demand* (or *query-driven*) mode, users decide when to gather data. They send instructions to the WSN indicating that they wish to receive data and then wait for the required type of data to be sent in the requested format. Users may even specify the future reporting periods; subsequent reports would then be sent in periodic reporting mode.

Target or event detection and tracking is a typical example of applications in event-driven reporting. Its purpose is to detect, classify, and locate specific targets or events, as well as track the targets or events over a specified region. Once there is an event or a target emerging in the area, the sensor nodes around the target or event gather the required information and report back to the sink. One characteristic of event-driven reporting is its real-time requirement. This means that data transmission latency is one of the key problems in these applications.

Targets can be divided into two categories: targets in the first category are individual objects that usually have a small size when compared to the sensing area of the network. These targets emit noise, light, and seismic waves, such that nearby sensor nodes are able to detect and track them. A typical example is to deploy a sensor network to detect troops, such as tanks and soldiers, in a battlefield. Once a tank moves into a specific area, information on the tank such as its location and speed, will be gathered by the sensor nodes and reported to the BS via multihop communication. The targets in the second category are continuous objects, which spread in the sensing area of the network. An example is the use of WSNs to detect and track diffused poison gas or chemical/biochemical liquids.

Figure 1.2 shows a typical scenario of event-driven reporting in WSNs. Sensor nodes are deployed in the sensor field to form a wireless network. Once there is an event in the monitoring field, such as a fire, a nearby sensor node, say *A*, will detect the fire if the sensed temperature exceeds a predefined threshold. Then *A* either starts the routing process (reactive) or uses the route in its routing table (proactive, e.g., *A*-*B*-*C*-*D*-*E*), to report information of the event to the sink. The sink may then take appropriate actions immediately or store the data in the database for future statistical use.

Periodic reporting is different from event-driven reporting. Data gathered in periodic reporting does not require urgent delivery to the sink. Further, the data in the event-driven reporting usually comes from sensors in the vicinity of a target or event, whereas the data in periodical reporting is normally gathered from sensor nodes throughout the sensor field.

Sensors report to the sink by applying *data gathering* and *data aggregation* operations. Data gathering refers to forwarding the measured data to the sink

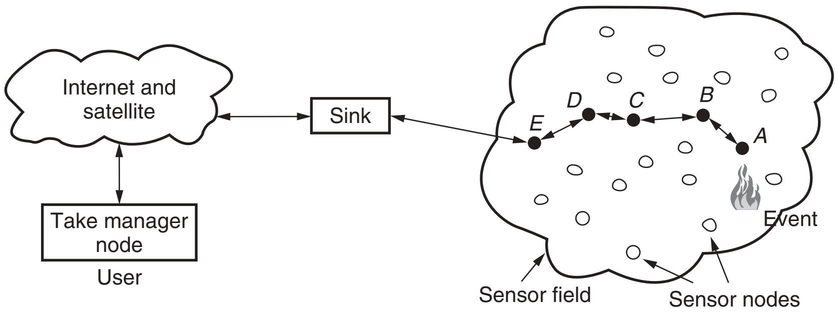


Figure 1.2 A scenario of event-driven reporting.

without further changes on the way toward sink. This is normally achieved via a *routing* task, that is, sending a message from a sender node (sensor) to a destination node (sink), using other sensors to forward the report. However, data collected by sensor nodes might be redundant, correlated, and/or inconsistent with data from other sensors. Data aggregation is used to combine data coming from different sensor nodes. This eliminates redundancy and minimizes the number of transmissions.

A general approach employed in data gathering and data aggregation is to construct a spanning tree which is rooted at the sink and connects all sensor nodes in the network. If one node fails, the topology will be reorganized into a new topology. Tree maintenance is usually an energy-demanding operation.

In data gathering or aggregation, data from each sensor is forwarded to the sink along the spanning tree. This is illustrated in Figure 1.3 where a WSN is deployed for agricultural applications. A large number of sensor nodes are scattered throughout a field to monitor the temperature, light levels, and soil moisture. The sink, located in the house, queries the sensors, which configure themselves. The reporting tree is constructed in the process and rooted at the sink. Data is periodically collected from all sensors in the field and sent to the sink. In data gathering operations, individual data from each sensor is forwarded along the tree without being combined with measurements from other sensors. Data aggregation may be applied too, for example, combining the readings from all of the sensors inside a zone and submitting a combined report via data gathering

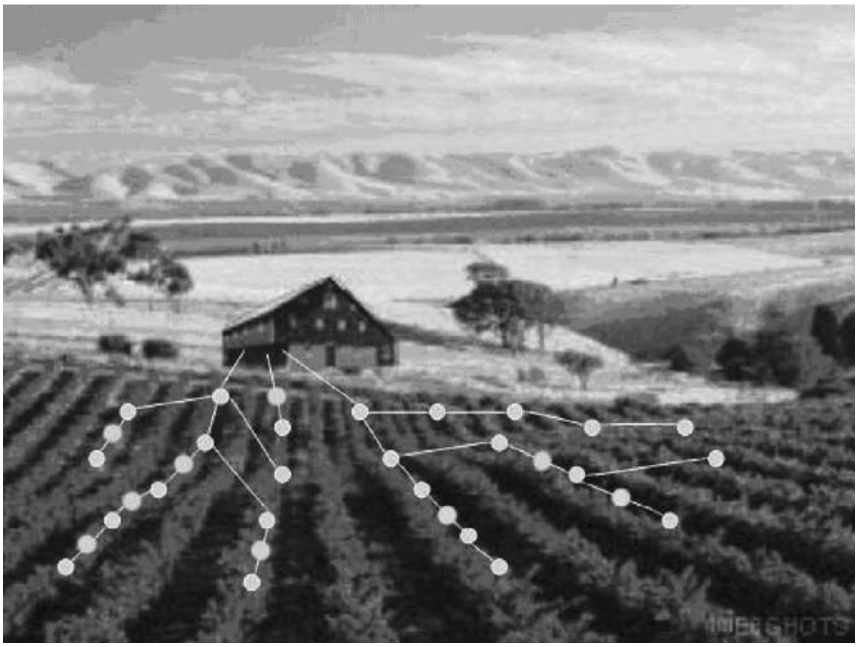


Figure 1.3 Data gathering or aggregation in agricultural applications.

operations along the tree hop-by-hop toward the root. For instance, information on soil conditions in different zones of the same field might be needed to apply uneven amounts of fertilizer. Sometimes a single report from the whole field would suffice such as information on the current temperature. In this case, upon receiving the data from each child node in the tree, a sensor node aggregates the data with its own before delivery to its parent node in the constructed tree.

The traditional view of large-sized static sensor networks with one fixed sink has been challenged by their theoretical and simulation analysis discovering some bottlenecks in their performance. For example, it was reported that while using the same transmission range for sensors optimizes energy per report (without data aggregation), it also creates *energy holes* around the sink while the periphery is left with almost full energy (Olariu *et al.*, 2006). Moreover, data aggregation is often impossible. For example, sensors monitoring movements do not generate the same reports and sink instructions are also not aggregated. Therefore, the problems do not seem to have a resolution unless the model itself is changed: It should be either small scale (e.g., up to hundred nodes) or involve multiple *sinks*, *mobile sinks*, *mobile sensors*, and so on. However, this in turn complicates network layer protocols.

1.5 UNIT DISK GRAPH MODELING, HOP COUNT METRIC, AND PROBABILISTIC RECEPTION

Multihop wireless communication in networks of equal devices applying same and fixed transmission radii (i.e., a homogeneous network), has a simple modeling that is an excellent and extremely useful simplification of the complex physical layer. In the *unit disk graph* (UDG), two nodes communicate if and only if the distance between them is at most R, where R is the transmission radius which is equal for all nodes. A UDG is therefore determined by the positions of nodes and a fixed common transmission range R. To illustrate this, if we use $R/2$ as the radius of the disk of each node, two nodes are connected if and only if their corresponding disks intersect. An example of a UDG is shown in Figure 1.4 below. Unit disk graphs successfully model WSNs, wireless *ad hoc* networks (used in rescue, conference, and battlefield scenarios), vehicular network communications, and wireless networks of actuators (to be defined shortly). In combined networks, such as sensor and actuator networks, they can model communication of each component network separately by using different transmission radii for them.

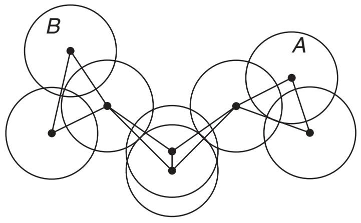


Figure 1.4 An example of a unit disk graph (with radius $R/2$).

Hop count can be used as a metric for routing in UDG if each node applies the same and fixed transmission power. It is defined as the number of hops from one node to another. Hop count between two adjacent nodes is 1. In Figure 1.4, the hop count between node A and node B is 4. In homogeneous networks, where nodes do not adjust transmission radius, the route with the smallest hop count from a source to a destination guarantees the minimal energy cost and the lowest transmission latency (assuming that the delay at each node is the same).

Although the protocols at the network layer are mostly designed with ideal UDG assumptions, experiments are normally carried out on simulators that implement more realistic physical and medium access control (MAC) layers. The UDG model is not realistic since variations in the received signal strengths are not considered. In fact, it has been pointed out that the impact of signal strength fluctuations is sometimes more significant than the impact of node mobility (Stojmenovic *et al.*, 2005a). Therefore, nondeterministic radio fluctuations cannot be ignored when designing robust protocols for sensor and *ad hoc* networks. In addition to distance, the received signal strength also depends on other factors such as environment and transmission medium.

Existing physical layer models, such as the combined Friis and two-ray ground model (Nadeem and Agrawala, 2004) and the lognormal shadowing model (Stojmenovic *et al.*, 2005b), require nodes to estimate the probability of receiving a bit or a packet based on either signal strength, distance between nodes, or merely by deriving statistics from a number of bits or packets recently sent between two nodes. The realistic physical model normally uses a function to represent the packet reception probability. For instance, the packet reception probability $p(x)$ in the shadowing model (Stojmenovic *et al.*, 2005b) depends on the length of the packet, and the distance x between two nodes. Suppose R is the distance so that the packet reception probability is $p(R) = 0.5$, the function $p(x)$ may have approximately the following values: $p(0) = 1$, $p(0.1R) \approx 1$, $p(0.5R) \approx 0.9$, $p(R) = 0.5$, $p(1.5R) \approx 0.25$, and $p(2R) = 0$. The values give a sufficient intuition on how to design physical layer–aware routing protocols. If a fixed signal-to-noise ratio (SNR) is assumed then the function $p(x)$ looks like the graph in Figure 1.5 (Kuruvila *et al.*, 2005). In this example, the probability for successful transmission at distance $d = 30$ is more than 0.95. If $d = 41$, the probability for successful transmission is around 0.5. This means that approximately half of the transmissions are successful. If $d = 50$, the probability for successful transmission decreases to around 0.05. Two nodes can still communicate as long as they make a sufficient number of attempts.

At the physical layer, the hop count metric may not properly reflect the real cost involved in a route. For example, suppose there are many long edges in the shortest path, in terms of hop count. Many retransmissions may be required between adjacent nodes on these long edges due to low probability of packet reception. Thus, the *expected hop count* (*EHC*) should be used instead. Expected hop count is defined as the expected number of messages between the sender and the receiver, including retransmission, acknowledgments and so on. Extended hop

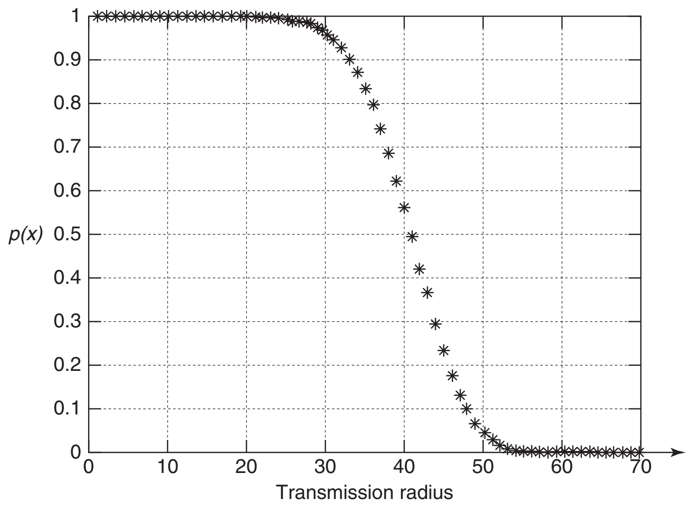


Figure 1.5 Packet reception probability versus transmission radius.

count measures can be also used to measure the cost of a route, by summing EHC values on each edge of the route.

Let S and D denote the sender and receiver, respectively. Suppose that acknowledgment is required for each successful transmission. Assume also that the sender repeats sending packets until it receives acknowledgment from the receiver. Let the distance between S and D be x, and the packet reception probability for transmission from S to D be $p(x)$. Thus, the probability that S does not receive any of the u acknowledgments from D is $(1 - p(x))^u$. That is, the probability that S receives at least one of the u acknowledgements from D is $1 - (1 - p(x))^u$. Therefore, $p(x)(1 - (1 - p(x))^u)$ is the probability that S receives an acknowledgment after sending a packet and therefore stops transmitting further packets. The total EHC between two nodes at distance x is:

$$1/[p(x)(1 - (1 - p(x))^u)] + u/[(1 - (1 - p(x))^u)],$$

where the first term is the message count and the second term is the acknowledgment count. In order to minimize the EHC, the value of u should satisfy $up(x) \approx 1$. That is, the best value of u for a given $p(x)$ is approximately $1/p(x)$ (Stojmenovic *et al.*, 2005b).

1.6 ADJUSTABLE TRANSMISSION RANGE AND POWER METRIC

Sensor nodes have the capacity to adjust their transmission ranges without incurring any significant cost for the adjustment. A common transmission radius is normally preferred because medium access protocols currently considered to be *de facto* standards, such as Zibgee, require it for proper functioning. However, finding a minimal common transmission radius is a nontrivial problem, especially for its maintenance. A possible compromise is that each sensor is made aware of its neighbors by using "hello" messages. But when the neighbor for forwarding is decided, the transmission power could be adjusted to reflect the distance. Here, we still assume the UDG modeling with adjusted transmission radius. In reality, there is an impact of the realistic physical layer at critical transmission distances, which will be discussed later.

A simple power consumption model is introduced in Rodoplu and Meng (1999). The total *power* needed to transmit and receive a message between two nodes at distance d is proportional to $d^\alpha + c$, where α is the signal strength attenuation factor which is normally between 2 and 5 depending on the transmission medium and the environment, and c is a contact that accounts for signal processing at the transmitter and receiver, as well as the minimal power to receive a signal properly. This model has been restated in Heinzelman *et al.* (2000). The energy consumption per bit is calculated as follows: $power = E_{\text{trans}} + \beta d^\alpha + E_{\text{rec}}$, where E_{trans} and E_{rec} are distance-independent terms which represent the overhead of the transmitter electronics and receiver electronics, respectively. For simplicity, E_{trans} and E_{rec} are often assumed to be the same (Chen *et al.*, 2004), and c from Rodoplu and Meng (1999) model is proportional to $E_{\text{trans}} + E_{\text{rec}}$. βd^α is the distance-dependent term that represents the power consumption required to transmit one bit from a sender to a receiver over distance d. If a message contains k bits, the power consumption is then normally multiplied by k.

When the transmission range of the nodes is adjustable, power metric is used to measure the optimality of a routing algorithm in different ways. The simplest way is to measure power consumption at each hop and look for a route that minimizes the total power consumption (the sum of powers consumed at each hop). However, some nodes may be centrally positioned and used on many paths. Their energies can be depleted while the energies of some peripheral nodes may remain close to their maximum. The lifetime of a network may be measured in several ways, including the moment the first node spends all its energy or network partitioning (the moment a particular sensor is not able to deliver its report to the sink because of energy holes in the network coming from sensors left with no energy). Thus, the minimum energy metric routing may not maximize the network's lifetime since some sensors may suffer early failure. An alternative is to maximize the lifetime of the network.

1.7 COST METRICS

A variety of metrics and their combinations can be used to design and evaluate communication protocols for WSNs. We have discussed so far hop count and power consumption metrics. A convenient metric that can be used to avoid nodes with low remaining energy on a routing path is called *reluctance* (Stojmenovic and Lin, 2001b). The remaining energy g at a sensor node can be normalized in the interval (0,1). The resistance f is then $f = 1/g$, meaning that the reluctance becomes huge when a sensor is close to depletion.

Some applications of WSNs, where real-time or multimedia data are involved in communications, require a guarantee on *QoS* metrics such as delay, throughput, and bandwidth. For example, sensor networks for fire detection require short latency to transmit emergency data to the sink. QoS routing is usually performed through resource reservation in individual nodes along the route.

In the sequel, the term *cost* will often be used to denote one of the mentioned metrics or a newly designed metric which is often a combination of several existing metrics. One example of a combined metric is *power * reluctance*, which can be used in designing routing paths to balance between finding routes with a low total sum of power metrics on route and also avoiding nodes with low remaining energy. The cost metric is therefore often used to find a trade-off among these parameters.

As another example, a conditional max–min battery capacity routing is studied in Toh (2001). If there is at least one route such that the residual energy of each node is greater than a specified threshold, the minimum energy metric is chosen. Otherwise, the route that maximizes the minimum residual energy is selected. In this algorithm, there exists a hidden cost for finding routes needed to elect the best one. The sender node needs global network knowledge to gather it, which requires communication overhead not accounted for in the selected metric of route efficiency. This point will be further discussed in the Section 1.15.

1.8 SLEEP AND ACTIVE STATE MODELING

Energy consumption is one of the key problems in WSNs. Several energy consumption models have been studied in the literature. The following discussion and the graph (Fig. 1.6) are based on the study by Barrenetxea *et al.* (2008). The graph shows energy consumption of a TinyNode sensor mote in different states. The experiment shows that the calculation of the sensor node's receiving costs depends on the assumption of the node's status. If it assumes that the radios of the nodes are always on, the energy consumption of the receiving costs is negligible since the cost for receiving packets has been included in the cost for keeping the radios on. More precisely, the energy consumption is equal to 2 mA when the radio is off but is equal to 16 mA when the radio is on for reception. This means that it takes about eight times more energy for listening compared to sleeping state. The total energy consumption for receiving depends on how long the radios need to be on to receive an incoming packet. Using the example in Figure 1.6 below, we suppose that transmitting a packet at 15 dB consuming 60 mA takes 5 ms. Receiving the packet takes at least 5 ms. However, it is not possible for the node to turn on exactly at the time the packet is sent. That is, to receive the packet, the node should turn on its radio for more than 5 ms (according to the used protocol). In Figure 1.6, the energy consumption of the radio is 15 mA. Therefore, if the total time the radio is on is more than 20 ms, the energy consumption of receiving a packet is more than the energy consumption of transmitting a packet.

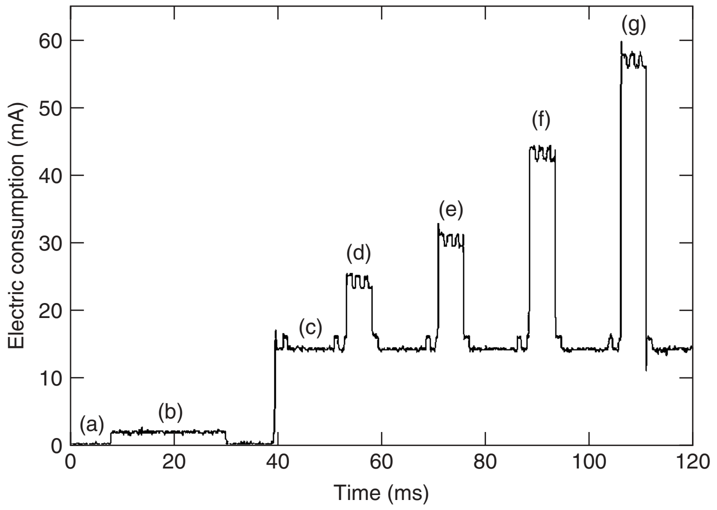


Figure 1.6 (a) CPU off; (b) CPU on; (c) radio on; (d) sending a packet at 0dB; (e) sending a packet at 5dB; (f) sending a packet at 10dB; (g) sending a packet at 15dB.

1.9 ARCHITECTURES FOR WIRELESS SENSOR AND ACTUATOR NETWORKS

Although WSNs have been employed in many applications, such as environment monitoring and health care, there are an increasing number of applications that require the use of actuators along with sensors. This occurs when the network system needs to interact with the physical system or the environment via *actuators* (also called *actors*). From the engineering aspect, an actuator is a transducer that accepts a signal and converts it to a physical action. Actuators transform an input signal into an action upon the environment. Typical examples of actuators are robots, electrical motors, and humans. Traditional sensor and actuator networks use wired communications among themselves; these networks have been well studied. The advent of small, intelligent, low-energy, and low-cost wireless sensing and actuation devices has the potential to significantly expand existing applications of wired sensor actuator networks. Wireless sensor actuator networks (WSANs) are emerging as the next generation of WSNs. The major difference between WSANs and WSNs is that WSANs are capable of changing the environment and physical world while WSNs cannot. Wireless sensor actuator networks are envisioned for applications that include disaster relief operations, intelligent buildings, home automation, smart spaces, pervasive computing systems, cyber-physical systems and nuclear, biological, and chemical attack detection (Xia *et al.*, 2007).

A WSAN usually consists of a group of sensor nodes that are used to gather information from the environment, and actuator nodes that are used to change the behavior of the environment. There are wireless links between the sensor and actuator nodes. Sensor nodes sense and report the state of the environment while actuator nodes gather data from sensors and are able to act on the environment. Wireless sensor actuator networks are expected to be self-organized and potentially operate autonomously in unattended environments, with basic and minimal directives from the user that might be remotely connected to the scene. In typical applications of WSANs, sensor nodes are static while actuator nodes, such as robots and humans equipped with vehicles, are mobile. However, some actuators such as sprinklers in fire detection systems, could be static. Sensor nodes also could be mobile in some scenarios (mobile sensors). For example, in sensor relocation problems, sensor nodes are required to move to locations of failed sensors for continued area coverage. Sensors and actuators can even be integrated into a single robot which is capable of sensing and moving. Compared to sensor nodes, actuator nodes usually have stronger capabilities in data processing, wireless communication, and power supply (Melodia *et al.*, 2007). Therefore, the number of sensor nodes deployed in a monitoring region may be in the order of hundreds or thousands while such size is not necessary for actuator nodes since they have higher capabilities and can act on larger areas.

Coordination is another aspect of WSANs (Akyildiz and Kasimoglu, 2004). Unlike WSNs, where the sink performs the functions of data collection and coordination, sensor-sensor, sensor-actuator, and actuator-actuator coordination is required in WSANs to achieve the overall application objective. *Sensor-actuator coordination* provides the path establishment for transmission of event data from sensors to actuators. This coordination may be also needed for some control traffic, such as communication locations of actors to sensors or helping sensors to learn their geographic position with higher precision. After receiving event data, actuators need to coordinate with each other to make decisions on the most appropriate way to perform actions. We refer to this process as *actuator-actuator coordination*. The coordination has additional aspects, such as fault tolerance, activity scheduling, and network design guidelines.

There are two basic architectures for data processing in WSANs described in Akyildiz and Kasimoglu (2004). One is called *automated architecture*, where sensor nodes sense the environment and report the data to actuator nodes which then initiate appropriate actions based on the received data. This architecture is shown in Figure 1.7a. The second architecture is called *semiautomated architecture* where sensor nodes route sensing data back to the sink which may then issue action commands to actuator nodes. This architecture is shown in Figure 1.7b. Semiautomated architecture is similar to the architecture of traditional WSNs. Therefore, current protocols and algorithms for traditional WSNs can be easily adopted in this architecture. The advantages of automated architecture are as follows. First, since sensing data is reported to actuators which are closer than the sink to sensors, communication latency is minimized. Second, in semiautomated architecture, transmitting the sensing data to the sink usually causes fast energy depletion of nodes which are around the sink. In automated architecture, sensing data is reported to actuators and different actuators may be triggered based on different events. Hence, the communication load can be more evenly distributed

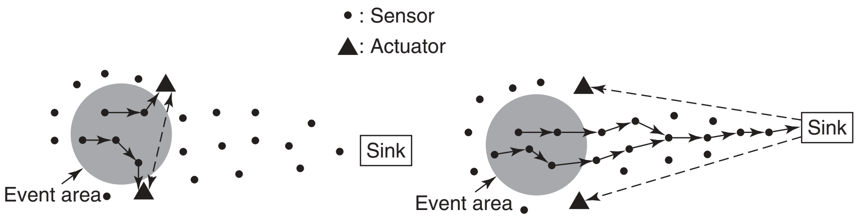


Figure 1.7 (a) Automated architecture and (b) semiautomated architecture.

among all nodes and it results in a longer lifetime of networks. Therefore, the automated architecture is able to provide low communication latency and longer network lifetime, which are desirable in most applications of WSANs.

The third architecture is proposed in Stojmenovic *et al.* (2007) and will be referred to here, as the *cooperative architecture*. In this architecture, sensor nodes transmit sensing data to actuator nodes via a single-hop or multiple hops. The actuators analyze the data and may consult the sink(s) before taking any action. That is, actuators may use their peer-to-peer network to make decisions and take action, possibly informing the sink about the action taken, or could inform the sink and wait for further instructions from the sink. A user (task manager) controls the network via the sinks. One or more of the actuators may also play the sink role. In fact, sinks can be treated as special kinds of actuators, although a better interpretation might be to associate them with BSs that communicate directly with the user. The architecture is illustrated in Figure 1.8, where one sink is linked to one of the actuators while the other actuators can reach the sink in a multihop actuator-actuator structure. Usually, actuators are more powerful and have a larger transmission radius than sensors. In extreme cases, actuators are able to directly reach all sensors in the network. Sensors route their data to any actuator. This task is known as the *anycasting* problem if a sensor is aware of the geographic positions of all actuators and itself. Sensors may start reporting

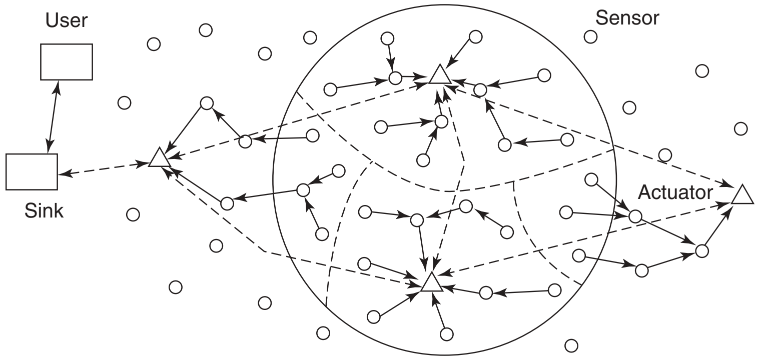


Figure 1.8 Cooperative architecture of sensor–actuator networks.

toward one selected actuator, but another actuator could be the ultimate receiver after some dynamic changes of the message destination. Alternatively, routes can be created by flooding from all actuators and memorized by sensors for reporting back toward the nearest sink.

The sink monitors the overall network and communicates with the task manager node and the sensor or actuator nodes. If necessary, the sink issues commands to actuators which forward them to sensors located inside the area that needs to be monitored (the circle in Figure 1.8). After sensors detect an event occurring in the environment, the event data is locally processed and transmitted to the actuators, which gather, process, and eventually reconstruct the event data. Coordination is one of the primary characteristics of WSNs.

Ruiz-Ibarra and Villasenor (2008) proposed a taxonomy for cooperation mechanisms in wireless sensor and actor networks. Wireless sensor actuator network frameworks consist of an architecture network (automated or semi-automated), a coordination level (sensor-sensor, sensor-actor, actor-actor), node mobility (fixed or mobile), and a network density (dense or sparse). Collaborative procedures include routing protocol, synchronization, localization, aggregation, clustering, encryption, power control (for event reporting and task execution), and QoS. The performance criteria include optimization criteria (metrics for reaching stated objectives), scalability, complexity order, and reliability (security, robustness). The application requirements consist of real-time constraints, event frequency, and concurrent events.

Some other architecture may be envisioned for futuristic applications. For example, Figure 1.9 shows a vision of merged *ad hoc* sensor and actuator networks in military applications. Sensors are placed in the field to detect minefields and firing locations, for target tracking, detecting chemical and biological attacks, and can be also attached to soldiers and vehicles. Vehicles, soldiers, and airplanes can also serve as actuators in the network.

1.10 SIMPLE MODELS AND APPLICATION OF WIRELESS SENSOR AND ACTUATOR NETWORKS

Current applications of WSANs rarely use theoretical models described in the previous section. There exists a gap between theoretical achievements and practice. We describe here several simple models for wireless sensors and actuators from recent literature, which do not really fall within presented classification. In all cases, the wireless communication is a single-hop one, direct communication between sensor and actuator or between two actuators.

A classical *star topology* of WSANs was studied in Korber *et al.* (2007). In the star topology, the BS serves as a network controller and as a gateway to upper layers. The BS may have a wired bus and a wireless radio interface. The TDMA (time division multiple access) technique is employed in the MAC layer. Each sensor is integrated into an actuator to form a sensor-actuator module. These modules are able to reach the BS in one-hop wireless communication. A time and frequency slot is allocated for each sensor and actuator, such that communication

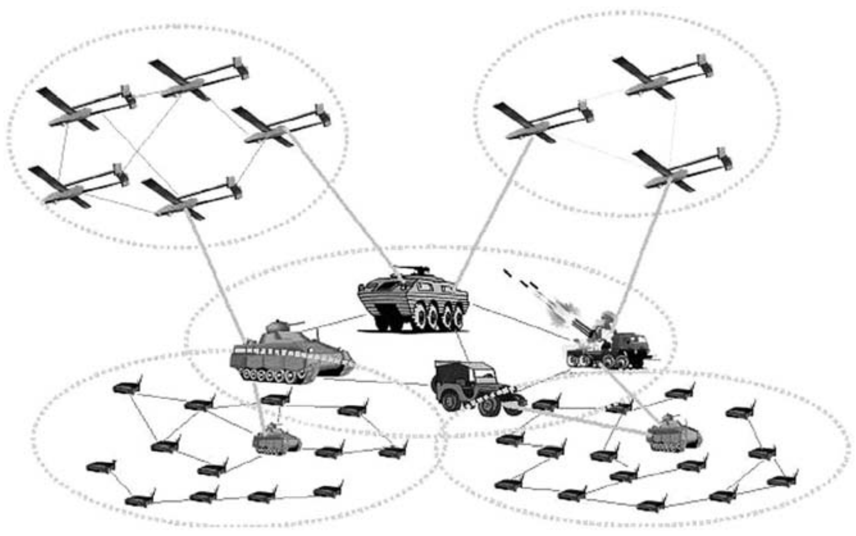


Figure 1.9 An example in military applications.

collisions between the sensors and actuators can be avoided. There is a trade-off between QoS, for example reliability and real-time communications, and the lifetime of nodes in WSANs. However, in many applications of WSANs, it is preferable to guarantee real-time communications and defined timing behaviors. The star topology is a good solution to satisfy the real-time requirement. The authors Korber *et al.* (2007) also argue that single-hop wireless communication is important for reliable industrial applications.

There is an application of WSANs in bull breeding paddocks, which is used to control the aggressive behavior of bulls. This application is described in Wark *et al.* (2007). Fighting between bulls during the breeding season may result in serious injuries to the bulls, which are high value animals. Therefore, it is critical for the breeding industry to protect these high value animals without human intervention. The idea is to deploy sensors and actuators in the cattle collars, which serve to detect and control the bulls' behavior. A hardware platform is capable of integrating a wide range of sensors and actuators, and it consists of an Atmega 128 processor and an 8MB flash memory. The onboard radio transceiver and hardware platform, along with the integrated stimuli board which acts as an actuator, are mounted inside a specially designed collar. The integrated sensor is able to estimate the dynamic states of the bulls from location and velocity observations. Once aggressive behavior in a bull is detected, the actuators initiate a stimuli on the bulls.

A sensor and actuator network in smart homes for supporting elderly and handicapped people was studied in Dengler *et al.* (2007). The primary goal was to monitor domestic systems such as air conditioning, lights, and heating, as well as to control the basic functions of the home entertainment and security systems. In the experiment, a test area which included "a living area" and "a kitchen" was used to represent the smart home environment. The sensor network consisted of three BTnodes, an autonomous wireless communication and computing platform based on a Bluetooth radio, and a microcontroller. Each BTnode was equipped with BTsense v1.1a sensor boards and proper sensors for light, motion, and temperature detection. Each sensor measured data at intervals of 30 s. If a sensor sent more than five unacknowledged emergency calls, the mobile robot was programmed to move directly to the sensor node. Another application for smart home monitoring has also been described (Li, 2006).

An example of the use of WSANs to monitor environments is the fire detection system. A group of sensor nodes are placed in a building or an area of interest. In the event of a fire in the monitoring region, the sensor nodes that are close to the origin of the fire report the location and intensity of the fire to water sprinkler actuators. On receiving alarm messages from sensor nodes, the water sprinkler actuators analyze the intensity of the fire and take appropriate actions before the fire becomes uncontrollable.

1.11 GENERATING CONNECTED WIRELESS SENSOR AND ACTUATOR NETWORKS

In this section, we will discuss the population of connected wireless sensor and/or actor networks, and how to generate one particular sample from the population for the purpose of simulating them and evaluating performance of proposed communication protocols. Typically, in literature, connected *random UDG* is employed in generating wireless sensor and/or actuator networks. It is generated by placing a group of nodes in a specific area pattern, such as a rectangle or a circle. The positions of N nodes are randomly determined (e.g., by selecting their two or three coordinates at random) and are independent from each other. The desired network topology is achieved once the generated topology passes the connectivity test (usually by running centralized Dijkstra's shortest path algorithm). The expected node *degree* (average number of neighbors per node) is the number of remaining nodes, $N - 1$, times the probability that any node will be placed within the node's transmission area. This probability can be approximately calculated by dividing the transmission area by the total area. Thus the expected degree, d is $\approx (N-1) \times \pi r^2/A$, where A is the area of the region of interest and r is the transmission radius. That is, $r \approx \sqrt{dA/(N-1)\pi}$. The exact average degree D for the generated graph is only an approximation of the desirable average degree d used in the graph generation.

It was observed that simulations in existing literature were using transmission radius r as the independent variable, while the corresponding average degree d was normally not even reported (Stojmenovic and Lin, 2001a). This has led to reporting simulation data for only dense graphs, for example, and hiding delivery problems for sparse graphs. To achieve the desired and accurate network density d, and use it as a parameter in simulations to study performance networks ranging from sparse to dense, Stojmenovic and Lin (2001a) proposed a method to control the average number of neighbors d by adjusting the corresponding common transmission range R. First, N nodes are generated at random. All $(N-1)N/2$ edges are sorted into a list in nondecreasing order. The transmission range R is set to be the length of the $(Nd/2)$th edge in the sorted list of all distance lengths. There is an edge between the two nodes if and only if their Euclidean distance is not greater than R. The generated network is then tested for connectivity.

The two generation algorithms, with approximate and accurate average degree as parameters, suffer from two problems. Since sparse networks have a high probability of being partitioned, they may generate a lot of disconnected topologies and take a long time before a connected UDG is obtained. Although it is reasonable to assume that the positions of nodes are independent in some scenarios, some networks may have characteristics different from random UDGs. For example, actuator nodes in WSANs create their own network to facilitate coordination and enhance data communications and actuation performance. This imposes certain restrictions on their locations with respect to each other. In some applications, the position of a newly deployed node may depend on the positions of other nodes that are already in the region of interest. For example, laptops of attendees in a conference form a multihop *ad hoc* network. When a new attendee with a laptop enters the conference venue, a good choice is to sit not very far from the others, so that the network service is available. At the same time, the new attendee may try to avoid overpopulated areas in order to have an acceptable throughput.

Fast generation of several types of wireless *ad hoc* networks where new node placement is dependent on other nodes' placements, has been studied (Onat *et al.*, 2008). Two classes of algorithms: rejection–acceptance and center node-based algorithms were proposed. In center node-based algorithms, for example minimum degree proximity algorithm (MIN-DPA), a center node is chosen among the already placed nodes before the placement of a new node. The new node is placed around the center node. In rejection- or acceptance-based' algorithms, a random candidate position is selected for each node during each round. Then, the position is either accepted or rejected depending on some constraints.

There are several constraints for placement of nodes. In the *proximity constraint*, a new node is placed at a minimum safe distance from any other existing nodes and should be closer than the approximate transmission radius from at least one of the existing nodes. In the *maximum degree constraint*, a new node position is rejected if its placement would make one or more of the existing or new nodes have a degree exceeding the maximal one set by a threshold. For sensor networks, *coverage constraint* is important. A candidate sensor position is accepted only if it sufficiently increases the overall coverage area. In the extreme case of aiming at full coverage, a candidate sensor is accepted if its coverage area is not fully covered by already placed nodes.

The basic idea of MIN-DPA is to place each new node around the center node that has the smallest degree in the current graph. The first step of the algorithm is to determine an approximate radius r, which is used to estimate degrees in the process. One of the existing nodes with the minimum degree is selected as the center node. A new node is uniformly and randomly placed within the transmission range of the center node (subject to possible boundary constraints). The procedure continues until all the nodes are placed. At the end, the approximate transmission radius r used in the generation process is replaced by the transmission radius R corresponding to the desired average degree d, using the "edge sorting" algorithm (Stojmenovic and Lin, 2001a) described above. After the placement, the connectivity of the topology is checked. Simulation results show that generating a connected graph using MIN-DPA is significantly faster than using UDG, especially for sparse networks.

In MIN-DPA, the position of a new node affects the degree of already placed neighboring nodes. MAX-DPA (maximum degree proximity algorithm) imposes a maximum degree constraint for all nodes in the network. In round i, a random position is chosen for node i, and accepted only if none of the existing or new nodes exceeds the maximum degree allowed $d_{\max}$. After all the nodes are placed, the transmission radius is adjusted and the connectivity of the topology is checked as in the case of MIN-DPA.

1.12 GENERATING MOBILE WIRELESS SENSOR AND ACTUATOR NETWORKS

A lot of research has been conducted on fixed sensor networks, where connectivity is normally demanded. However, a number of applications require mobile sensors. Mobile sensors and mobile actors may not preserve connectivity and often the application itself involves sporadic connectivity. Examples include vehicular networks where cars can be seen as sensors carrying information or actuators with possible actions such as changing speed, lanes or roads. People or wild animals can also act as actuators.

The model based on social network theory (Musolesi and Mascolo, 2007) views networks as collections of disconnected clusters. Each cluster is a connected network and nodes may move occasionally from one cluster to another (social movement), according to attractive "virtual forces" from other clusters. This is illustrated in Figure 1.10.

Some applications are based on harsh environmental conditions, such as underwater sensor networks of seals. In this application, batteries are impossible to change and could be lost since seals change their fur periodically. Networks can be very sparse. Seals can meet in clusters but then, they rarely meet at sea. There is no human pattern of day or night behavior.

It is worthwhile to mention that a collection of real mobility traces in various wireless networks is maintained at http://crawdad.cs.dartmouth.edu/.

1.13 PROBLEMS AT PHYSICAL, MAC, AND TRANSPORT LAYERS

Since a WSAN can be treated as a union of a WSN and an actuator network (mobile *ad hoc* network), current problems and challenges with sensor networks and *ad hoc* networks also exist in WSANs.

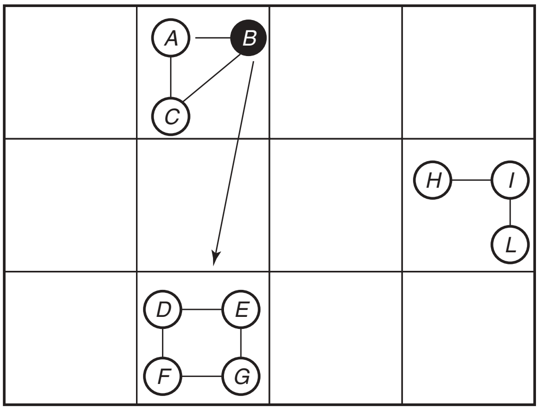


Figure 1.10 Social clustered network and social movement.

At the *physical layer*, it is required to process signals, deal with the hardware failure of sensor nodes, manage limited bandwidth and limited power, control sensing range and transmission range, and select antennas and operating channels. Energy scavenging and nontraditional power sources are surveyed in Roundy *et al.* (2005). Sensors use wireless communication, where RF noise and multipath fading causes severe packet losses. It is easy to eavesdrop and to launch spoofing or Denial-of-Service attacks. Infrared and optical lines of sight are alternatives that are considered. Nodes are at physical risk because they can be defective, lost, damaged, compromised or can have expired. Wireless communication implies limited bandwidth and in most cases also limited power (unless rechargeable battery, solar power or other energy supply alternatives are feasible). Wireless communication also implies one-to-all communication where messages sent by one node are simultaneously received by all neighbors within transmission radius. Smart omnidirectional antennas can also be considered (especially for actors) but this makes sensors more complex. Small processing power limits processing time and reduces the choices available for security solutions, data compression, and error control techniques. Routing table sizes are small due to reduced memory size and also reduced usefulness of such tables. Sensors are normally assumed to be all on the same frequency (or perhaps two frequencies, one being used for some control messages), since otherwise lots of communications sent on a "wrong" frequency can cause significant loss of energy and also the inability to find neighbors in a timely manner, during the time neighbors are active. Thus, frequency-hopping solutions like Bluetooth are currently not considered feasible for WSANs.

The *MAC layer* primarily aims at energy-efficient and collision-free communications. Medium access in WSNs is currently envisioned via IEEE 801.15.4 standard (IEEE Standard 802, 2003) and its extension known as Zigbee. ZigBee network specification (ZigBee Alliance, 2004) is one of the first standards for *ad hoc* and sensor networks. ZigBee is a specification for a suite of high-level communication protocols using small, low-rate, and low-power digital radios for wireless personal area networks (WPANs). The technology is intended to be simpler and cheaper than other WPANs, such as Bluetooth. Two network topologies are allowed by the standard, both relying on the presence of a central coordinator. In the peer-to-peer topology sensors (devices) may communicate directly, while in the star-shaped topology they must communicate through a coordinator. In a typical ZigBee network, the network addresses of nodes are organized in a hierarchical manner, such that one node can easily identify the addresses of its tree neighbors, for example its parent and children. A coordinator buffers all the packets for its associated devices, such that devices can go to sleep mode and wake up only when they need to retrieve data from the coordinator. Moreover, the coordinators route all packets for the devices. One can observe that the roles of coordinators and devices are similar to the roles of actuators and sensors, respectively. Sensors are time synchronized and follow a joint sleep–active schedule. They are active at the same time, followed by longer sleep periods. At the beginning of active periods, they compete for upcoming slots to send messages. While sensors are all active, Zigbee medium access works similar to the popular WiFi standard based on IEEE 802.11. Time is slotted. A station that has a message to transmit will first wait for a few slots of interframe separation. It generates a random number x in a certain interval (e.g., [0.31]) and uses carrier sense to determine whether or not the channel medium was used in each slot while waiting for retransmissions. The station will wait for x idle slots (without transmissions from any station) and afterwards will transmit the full message without further verification of a possible collision. Some other proposals that deviate from random access-based Zibgee were also considered. For example, Z-MAC (Rhee *et al.*, 2005) combines TDMA and CSMA. It switches MAC to CSMA and TDMA when contention is low and high, respectively.

At the *transport layer*, data gathering and data aggregation is scheduled in order to reduce traffic, increase reliability, and provide QoS control. Most of these problems are well-studied in WSNs and *ad hoc* networks. Note that traditional end-to-end reliability in wired networks is not applicable in wireless networks since link failure is possible due to the mobility or energy depletion of nodes. Thus, the QoS issues in WSNs usually call for reliability of communications rather than bandwidth and/or delay. Individual sensor measurements are not reliable and need to be combined with readings from other sensors to achieve collective reliability. The primary task in the transport layer in WSNs is, in fact, to achieve a sufficient level of reliability in the sensor network reports to the sink while minimizing their energy and bandwidth resources.

Wireless sensor networks and WSANs have a number of additional issues, often of a cross-layer nature, each of which is covered in literature and is important for overall network functioning. They will not be investigated in this book, which instead concentrates on basic network layer problems. The reader is advised to consult some other sources, such as handbooks on sensor networks (Stojmenovic, 2005), which cover authentication, key management, security issues, operating systems, databases, path exposure, target location, classification, tracking, data gathering and fusion, localization (position determination), time synchronization, and calibration.

1.14 PROBLEMS AT THE NETWORK LAYER

Current problems at the network layer can be classified into three categories: topology control, routing, and coordination. Their coverage follows.

1.14.1 Topology Control

A well-organized network topology can not only prolong the lifetime of a network, but also enhance data communications. Topology control problems can be subdivided into neighbor discovery problems and network organization problems. Neighbor discovery problems are defined as problems in detecting and discovering neighbors which are located within the transmission range. In the network organization problems, each node chooses its neighbors and constructs local topology by either adjusting its transmission power or setting its status, such as sleep and active modes. There are some protocols that achieve desired network topology by movement control on nodes. For example, the localized mobility control protocol in Das *et al*. (2007) constructs a biconnected network from a connected network through the movement of nodes. In this and a number of other protocols, topology control is used to create fault-tolerant networks for reliable communication protocols.

The most important topology control, especially for power-critical sensor networks, is to place as many possible sensor (and similarly actor) nodes into a sleep mode as possible. All nodes that are not essential for communication or area coverage can be placed in sleep mode for prolonged periods, synchronously or asynchronously. This is in addition to synchronized sleep–active state changes of currently active sensors for power efficiency at the MAC layer. There are a number of studies on energy efficiency at the MAC layer, which are also based on topology control. S-MAC (Ye *et al*., 2004) divides nodes into clusters based on fixed common sleep schedules to reduce control overhead and enable traffic-adaptive wake-up. T-MAC (Dam and Langendoen, 2003) extends S-MAC by adjusting the length of the waking time of the nodes based on the communication of neighboring nodes. B-MAC (Polastre *et al*., 2004) employs an adaptive preamble sampling scheme to reduce the duty cycle and minimize idle listening.

Some topology control schemes aim at selecting certain nodes from the network to create a *backbone* that can be used in several ways. A backbone is connected if the network of solely backbone nodes remains connected (after selecting them from the originally connected network). Some backbone structures are used to improve the efficiency of data communication protocols. For example, routing or broadcasting remains successful if intermediate nodes are selected only from connected backbones since each nonbackbone node has a neighbor from the backbone. Another possible application of the backbone set is to place the remaining nodes into sleep mode.

Clustering and *connected dominating sets* (CDSs) are two basic techniques used to generate the backbone for wireless sensor and *ad hoc* networks. The clustering process divides the nodes of a network into several clusters. In each cluster, there is a *clusterhead*, which is responsible for the coordination and data communication between nodes in the cluster. The selection of clusterheads is done via global nomination or local election, according to a certain protocol. Communications within a cluster could be one hop or multihop. The backbone could contain only clusterheads or may include some gateway nodes to enable connectivity.

Dominating sets are another technique for backbone creation. A subset of the vertices of a graph is called a *dominating set* if every vertex in the graph is either in the subset or is adjacent to at least one vertex in the subset. A CDS requires also connectivity among the backbone nodes. In the example of Figure 1.11 below, subsets {1, 2, 3, 5, 6, 10} and {4, 7, 8, 9} are dominating sets. The subset {4, 7, 8, 9} is also CDS while the former one is not.

Topology control may also be applied to select certain existing edges of the network while ignoring others. This leads to subgraphs that may have useful properties. For example, the Gabriel graph is a planar subgraph of UDG which can be used to guarantee delivery in position-based routing without relying on any memorization (Bose *et al.*, 1999).

1.14.2 Data Communication

In data communication problems, such as routing, QoS routing as well as multicast, broadcast, and geocast, the primary goal is to fulfill a given communication task successfully between nodes in the network. It requires, at the same time, the minimization of communication overhead and power consumption.

Routing is one of the critical issues in almost any type of network. It is used to find a route from a source to a destination in the network. In WSANs, each of the source and destination nodes could be either sensor or actor. In *QoS routing*, selected routes should satisfy the QoS criteria such as delay and/or bandwidth for real-time and multimedia-rich data communications.

In a *multicasting* task, the same message needs to be routed from a source node to a fixed number of *k* known destinations. *Broadcasting* is a special case

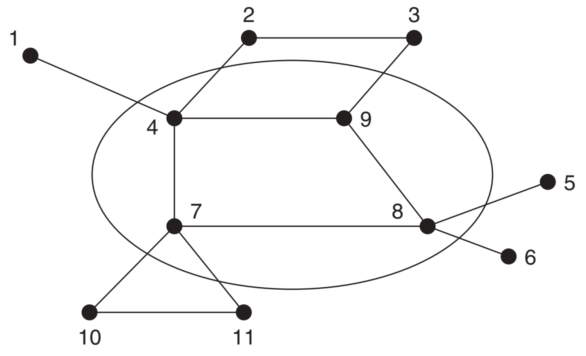


Figure 1.11 An example of a connected dominating set.

of multicasting where $k = N$, the number of nodes in the network. That is, in the broadcasting task the message is to be sent from one node to all the other nodes in the network. In a sensor and actuator network, multicasting is usually applied from a sensor node to send the same report to a fixed set of actuators. In some applications, such as monitoring a certain area, geocasting operation is carried out. In *geocasting*, the source sends messages to all the nodes located in a particular geographic region. For example, an actuator may request all sensors located in a certain region to report sensed movements.

Other basic data communication primitives may be needed in particular scenarios. For example, sensors monitoring certain movements may be required to continuously send video images of the object. At the same time, the actuator or sink may need to improve the quality of the delivered image by improving the communication links along the route. In this scenario, the traffic is large enough in volume and duration to warrant nodes expending energy on movement in order to forward the large traffic in a more efficient manner. Such a mobility control routing task and an algorithm were proposed in Liu *et al.* (2007). The primary goal was to move toward a route with minimal total power consumption while preserving communication during mobility, and achieving this with small total movement distance of actors or mobile sensors involved in routing.

1.14.3 Coordination

Wireless sensor actuator networks require coordination not only among sensors or actuators, but also between them. To facilitate the coordination, the first problem is to achieve proper actuator selection. Sensors need to know where and how to send reports to the closest actuator. Actuators may flood the network with their position or merely their IDs. Sensors receiving such information may rebroadcast them (so that more sensors learn about the same actuator) or ignore them if, for example, a closer actuator was already identified. Such flooding type algorithms from multiple actuators have been discussed in Ingelrest *et al.* (2006). Similar to flooding of route discovery in routing, sensors may learn their paths toward the nearest actuator, in terms of hop count or other metric distance. If position information is available and used in routing, the path may be found dynamically. When a sensor has a data report for actuators, it needs to efficiently find the best actuator to deliver the report, without flooding all the actuators. *Georouting* to the geographically closest actuator is an option, but it is often not the optimal one if, for instance, there is a void area between the sensor and that actuator. Instead, it may be more efficient to try to find a route to any of the actuators. Routing may start toward the physically closest actuator, but the destination actuator may be changed during the path search. This task is known as *anycasting*. The actuator selection problem also exists in the clustering stage of sensor and actuator networks. Each sensor needs to find available actuators and decide to join the cluster which is dominated by an actuator.

Sensors usually report to the sink or actuators via hop by hop transmission. However, mobile actuators are able to move to collect reports periodically via a predesigned route and thus minimize power consumption of sensors. For example, the U.S. Marine Corps used an unmanned airborne vehicle (UAV) to drop several sensor motes to detect vehicles traveling through an isolated desert area (Hill, 2003). The motes organize themselves to construct a network to monitor moving vehicles and record tracking information. The UAV plane comes back later and retrieves each node's tracking data.

There are several coordination problems associated with *location service* in WSANs. In location service problems, actuators need to provide and maintain (if they move) position information for sensor nodes and sensors need to maintain position information for the nearest actuator or neighboring actuators. Sensor-sensor, sensor-actor and actor-actor coordination may be used to provide position information to some sensors, using position information of nearby actors and possibly some "landmark" sensors in the field, as well as collaborative processing of neighborhood graphs.

In the *sensor relocation* problem, mobile actuators or mobile sensors move to replace failed sensors. Similar problems include coordinated movement of sensors to place themselves strategically around the point of interest, for efficient monitoring *(focused coverage* problem).

1.15 LOCALIZED PROTOCOLS AS THE SOLUTION FRAMEWORK

On the basis of the information required to run algorithms, existing algorithms or protocols in wireless networks can be roughly classified into two groups: *globalized protocols* and *localized protocols*. In globalized protocol, one or more nodes (usually the central node like a BS) need(s) to gather global information ranging from detailed information such as the whole network topology to simple information of global nature, such as the maximum degree in the network, to execute the protocol. However, in localized protocol, each node makes protocol decisions based solely on some local knowledge available. To be more precise, local knowledge in wireless networks is based on information from neighbors within k hops from a certain node, where k is usually a small integer like 1 or 2. Some protocols, for example beaconless georouting, do not require any information from neighbors. Another such example is the probabilistic flooding scheme (Ni *et al*., 1999) where each node makes its own decision about possible retransmission using a predefined fixed probability.

Shortest (weighted) path routing is a typical globalized protocol which computes the best route between the source and the destination. Both well known shortest weighted path algorithms, Dijkstra's and Prim's, require all nodes to know full network topology (all nodes and all edges between them) to make proper decisions. Globalized shortest path routing has huge setup costs because of the large number of messages exchanged in order for the source to gather the network topology. Global information is costly to gather and maintain in wireless sensor, actuator, and *ad hoc* networks (unless the network is static and small scale or single-hop). The dynamic nature of these networks (possible mobility and changes in activity status, arrival, and departure of nodes) requires localized protocols so that communication overhead needed for gathering global information is avoided. When the network size becomes large, performance of globalized protocols is degraded dramatically compared to localized algorithms. If scalability is an important protocol requirement in wireless networks then localized protocols are the best choice. In localized protocols, decisions are made based only on information from neighbors and natural additional information. For example, greedy routing (Finn, 1987) is a localized protocol which uses only position information of one-hop neighbors and the destination. In greedy routing, the current node on the route selects the neighbor that is closest to the destination.

In many cases, the global nature of described protocols appears hidden. One example is the well-known Bellman–Ford algorithm, used for Internet routing, which also finds shortest path routes. Neighboring nodes periodically exchange their routing tables that contain the costs and forwarding neighbors for each destination. Routing tables are then updated after each such exchange. While this appears to be a localized algorithm because of exchange between neighbors as the only apparent communication, a repeated application of that operation involves the arrival of important information from distant nodes in summary form. Therefore the algorithm is globalized. Some other algorithms are claimed to be localized in the literature although the message complexity for each node is not bounded (it is often O(d), where d is the number of neighbors). While the required information appears again to be strictly localized, the information gathered may still be of global nature in the process and the algorithm is not localized.

Localized protocols can be further classified based on the amount of information required and the overhead in the construction and maintenance phases. The amount of required information is related to the message complexity which is defined as the average number of messages exchanged per node in the protocol. In the construction phase, some localized protocols may need extensive message exchanges among neighbors. The amount of messages may then be comparable to that in the collection and use of global information. In the maintenance phase, some localized protocols (in the construction phase) may require message propagation and recomputation of the entire network for a change only in one part of the network, such as maintenance of a minimal spanning tree (MST) and a typical clustering structure. Thus, localized protocols can be further classified into *local localized* (message complexity in the maintenance phase remains low) and *quasi-local localized* (local changes may trigger global updates). Mobility of nodes and status changes between active and sleep modes require localized algorithms, preferably local localized.

Note that the expressed criticism for globalized algorithms does not mean that they are not useful for protocol design in scalable sensor and actuator networks. For example, globalized algorithms can be often applied with limited local knowledge (e.g., two hops), leading to winning protocols in several cases. This will be elaborated later in this book.

1.16 IMPLEMENTATION OF SENSOR MOTES

Many different versions of wireless sensor devices (also called *motes*) have been designed and built by various companies and institutions. The size of these motes varies from the size of a box of matches to the size of a pen tip. The smallest sensors are known as *smart dust*. We now describe several representative sensor motes.

The MICA mote is built by Crossbow in the United States. It consists of the Atmel Atmega 103L processor which is capable of running at 4MHz, has a 128kB flash memory, a 512kB serial flash, 4kB SRAM and a 4kB EEPROM. The MICA mote is powered by two AA batteries and the lifetime is up to 1 year under very low duty cycles. The mote operates at 916MHz or 413MHz and the transmission rate is 40 kbps with a transmission range of 100 feet. The MICA2 mote is the next generation commercial mote by Crossbow. It has the same processor and memory as the MICA mote but the radio transceiver operates on 433MHz or 868/916MHz with a transmission rate of 38.4 kbps. The outdoor range of the MICA2 mote is up to 500 feet. Both the MICA mote and MICA2 mote use TinyOS, an open-source embedded operating system developed at University of California, Berkeley, to control the mote and its attached sensors. The MICA2 motes accept the same sensor boards as the MICA mote.

Intel developed a mote in which the original modular design of the Berkeley motes are maintained while the data processing and battery life are improved. The Intel mote consists of a powerful ARM processor, SRAM, and flash memory. Optimal sensor boards and an optional power regulator are available. It is also based on TinyOS. The software stack includes an Intel Mote-specific layer with Bluetooth support and platform device drivers, as well as a network layer for topology construction and multihop routing. Security features, such as authentication and encryption, are also provided.

Sun SPOT (Sun Small Programmable Object Technology) is a sensor developed by Sun Microsystems (it appears to be the recommended choice in 2008). It can be battery- or USB-powered and is built upon the IEEE 802.15.4 standard. Sun SPOT is able to host a wide number of add-on boards with USB, TWI, SPI, I2S, RMII, USART and SD/MMC interfaces (https://spot-espot.dev.java.net). The most significant feature of Sun SPOT is that it runs Squawk Java Virtual Machine (VM) without an underlying OS. This VM acts as both, an operating system and software application platform. Moreover, Sun SPOT is a completely open source technology based entirely on Java technology. The open source release of the Sun SPOT platform includes hardware architecture, software, and the VM. The Squawk VM is the only open source research VM that is Java Logo certified.

Besides sensor motes, there exist integrated sensor and actuator nodes, such as robots, which are designed by several robotics research labs. For example, low-flying helicopter platforms provide ground mapping and air-to-ground cooperation of autonomous robotic vehicles (Thrun *et al.*, 2003). Autonomous battlefield robots sponsored by the Defense Advanced Research Projects Agency are able to detect and mark mines, and carry weapons. The robots developed by

Sandia National Lab may be the world's smallest autonomous robots. They are only 0.25 cubic inch and weigh less than an ounce (Akyildiz and Kasimoglu, 2004).

1.17 EXPERIMENTS ON TEST BEDS

Wireless sensor networks have been implemented on test beds in applications for environmental monitoring, business, military, health care and so on. A pioneering work is the use of a WSN for habitat monitoring on Great Duck Island (Mainwaring *et al.*, 2002). MICA motes are adopted as sensor nodes. The MICA Weather Board provides sensors which are able to monitor changing environmental conditions with the same functionality as a traditional weather station. The MICA Weather Board includes temperature, photo resistor, barometric pressure, humidity, and passive infrared sensors. Thirty-two motes were deployed on the island for 4 weeks.

A recent experiment on environmental monitoring used a flock of micro air vehicles (MAVs) to sense weather phenomena (Allred *et al.*, 2007). Each MAV may be equipped with temperature, pressure, humidity, wind speed or direction and/or other sensors. The MAVs are able to provide detailed mapping of hurricanes, thunderstorms and tornados, and also return data to ground stations. These data are useful in improving storm track predictions and in the understanding of storm genesis and evolution. In the experiment, the MAV is designed to keep the weight and the maximum speed of the airplane under 500 g and 20 m/s, respectively. The CUPIC autopilot board is employed. It contains a CPU, pressure sensor, radio, rate gyro and GPS device that send navigation information to the CPU. The cost of the entire airplane is less than $600. Five MAVs are employed in the experiment. An XBee Pro Zigbee class 2.4GHz radio is used to support both air-to-air and air-to-ground wireless communications. The MAVs are always operated at an altitude of less than 150 m to avoid potential conflict with larger airplanes and to maintain communication with remote control pilots.

A heterogeneous architecture for light monitoring and control was studied in Li (2006). It consists of six to eight light-sensing nodes and several actuator nodes that are connected to dimmers. A sensing network and an actuation network operate separately but are joined at a central gateway. The sensors, which match required conditions, reply to the gateway. The condition could be "sensors for which the light reading is higher than 4000" or "sensors located in the living room". The gateway then sends command messages to actuator nodes to control the lights.

The energy company BP employed motes on the Loch Rannoch, a big oil tanker, to predict failures of onboard machinery. One hundred and sixty motes were placed near some of the ship's equipment to measure vibrations in the ship's pumps, compressors, and engine as an indicator of potential failure. The system initiates an alert if unusual vibration or motion is detected. The experiment demonstrates that motes with relatively low cost are able to help protect expensive machinery (Steel, 2005).

Wireless sensor networks could be integrated into other networks, such as Internet and 3G networks. A video surveillance system which is composed of 3G, Internet, and WSNs was studied in Tso *et al.* (2007). The system consists of five components: a sensor network with a sink, a 3G phone-controlled patrol robot, a 3G handset, a laptop connected to the Internet, and a central gateway. The sensor network is used to detect abnormal events or intruders and report the sensing data to the central gateway via the sink. The central gateway analyzes the data and, if necessary, automatically sends an SMS notification to a user. The user can dial and instruct the robot to patrol on-site at a specific location to retrieve the real-time video via a 3G phone or Laptop. In the experiment, four sensors are deployed in the corners of the inspected room. The abnormal event is artificially set to the change of light intensity.

1.18 EXPERIENCES WITH THE DEVELOPMENT OF SENSOR NETWORK SYSTEMS

In Tanenbaum *et al.* (2006), authors argue that building sensor systems is a challenging task by discussing several considered scenarios. Monitoring Mexico's borders will be slow and costly for sensors as well as humans nearby. Sensors dropped in enemy territory need to be close to each other (sensing range about 10 m), therefore having a soldier watching the same area might be much more productive. Sensors detecting fire in a forest may not be able to deliver the report because of lifetime issues. Sensors need to be lifted for increased radio range. Sensors placed to monitor certain small regions can be easily activated with false alarms (e.g., by intentionally sending animals nearby).

In Barrenetxea *et al.* (2008), an efficient and cheap out-of-the-box environmental monitoring system is described. It is a time-driven network where sensors report environmental data (wind speed and direction, soil moisture, temperature, humidity, radiation, precipitation, etc.) to a sink, which in turn relays data to a publicly available database server. A sensing station consists of a four-legged skeleton containing a sensor box (containing a sensor mote as well as primary and secondary batteries) and a solar panel. Close to 100 such stations were deployed in the largest system. The communication stack consists of application, transport, network, and MAC layers as well as a radio medium. The application layer only queries sensors and batteries, and passes data to the transport layer. The transport layer does not include any congestion avoidance mechanism. It creates data or controls packets with 4 bytes of network header (containing hop count, sender ID, cost to sink, and sequence number) and 24 bytes of application payload. The network layer passes packets to the MAC layer. The MAC layer manages the radio and sends or receives packets. It is based on a simple backoff mechanism without carrier sense. The neighborhood is managed by beacon messages initiated from the sink. Each sensor updates its cost (only hop count was used) to reach the sink. The link quality is estimated by the ability of a neighbor to receive a data packet and to forward it. The time synchronization is achieved by a similar flooding initiated from the sink. Its frequency is decided to just offset the time-drift in sensors. The power management is resolved by duty cycling, where all nodes are synchronously sleeping and waking up, and with messages not starting before maximum time-drift following wake-ups. Routing is opportunistic, that is, a message is sent to any neighbor with a smaller hop count and decided at random at forwarding time. This ensures load balancing.

REFERENCES

Akyildiz IF, Kasimoglu IH. "Wireless sensor and actor networks: research challenges". Ad Hoc Netw 2004;2:351–367.

Allred J, Hasan AB, Panichsakul S, Pisano W, Gray P, Huang J, Han R, Lawrence D, Mohseni K. "SensorFlock: an airborne wireless sensor network of micro-air vehicles". Proceedings of SenSys'07; 2007. pp. 117–130.

Barrenetxea G, Ingelrest F, Schaefer G, Vetterli M, Couach O, Parlange M. "SensorScope: out-of-the-box environmental monitoring". The ACM/IEEE 7th International Conference on Information Processing in Sensor Networks (IPSN 2008); St. Louis (MO); 2008 April 22–24.

Bose P, Morin P, Stojmenovic I, Urrutia J. "Routing with guaranteed delivery in Ad Hoc wireless Networks". Proceedings of 3rd ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL M99); 1999. pp. 48–55.

Chen WP, Hou JC, Sha L. "Dynamic clustering for acoustic target tracking in wireless sensor networks". IEEE Trans Mobile Comput 2004;3(3):258–271.

Dam TV, Langendoen K. "An adaptive energy efficient MAC protocol for WSNs". Proceedings of ACM Sensys; 2003.

Das S, Liu H, Kamath A, Nayak A, Stojmenovic I. "Localized movement control for fault tolerance of mobile robot networks". Proceedings of the First IFIP International Conference on Wireless Sensor and Actor Networks (WSAN 2007); Albacete, Spain; 2007 Sept 24–26.

Dengler S, Awad A, Dressler F. "Sensor/Actuator networks in smart homes for supporting elderly and handicapped people". Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07); 2007.

Finn GG. "Routing and addressing problems in large metropolitan-scale internetworks". Technical Report ISI/RR-87-180: Information Sciences Institute (ISI); 1987.

Heinzelman WR, Chandrakasan A, Balakrishnan H. "Energy-efficient communication protocol for microsensor networks". Proceedings of 33rd Hawaii International Conference System Sciences; 2000.

Hill JL. "System Architecture for Wireless Sensor Networks" [PhD dissertation]: UC Berkeley; 2003, pp. 155.

IEEE Standard 802. "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)"; 2003.

Ingelrest F., Simplot-Ryl D, Stojmenovic I. "Routing and broadcasting in hybrid ad hoc and sensor networks". In: Wu J: editor. Theoretical and algorithmic aspects of sensor, Ad hoc wireless and peer-to-peer networks. Auerbach Publications (Taylor & Francis Group); 2006. pp. 415–426.

John M, Fulford-Jones TRF, Bonato P, Welsh M. "A wireless, low-power motion analysis sensor for stroke patient rehabilitation". Abstract 143281, Biomedical Engineering Society (BMES) 2005 Annual Fall Meeting; Baltimore (MD); 2005 Sept 28–Oct 1.

Korber HJ, Wattar H, Scholl G. "Modular wireless real-time sensor/actuator network for factory automation applications". IEEE Trans Ind Inform 2007;3(2):111–119.

Kuruvila J, Nayak A, Stojmenovic I. "Hop count optimal position-based packet routing algorithms for ad hoc wireless networks with a realistic physical Layer". IEEE J Sel Areas Commun 2005;23(6):1267–1275.

Li S-F. "Wireless sensor actuator network for light monitoring and control application". Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC 2006); 2006. pp. 974–978.

Liu H, Nayak A, Stojmenovic I. "Localized mobility control routing in robotic sensor wireless networks". Proceedings of the 3rd International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2007), LNCS 4864; Beijing, China; 2007 Dec 12–14.

Mainwaring A, Polastre J, Szewczyk R, Culler D, Anderson J. "Wireless sensor networks for habit monitoring". Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications; 2002. pp. 88–97.

Melodia T, Pompili D, Gungor VC, Akyildiz IF. "Communication and coordination in wireless sensor and actor networks". IEEE Trans Mobile Comput 2007;6(10):1116–1129.

Musolesi M, Mascolo C. "Designing mobility models based on social network theory". Mobile Comput Commun Rev 2007;11(3).

Nadeem T, Agrawala A. "IEEE 802.11 Fragmentation-aware energy-efficient ad-hoc routing protocols". Proceedings of the First IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS); 2004. pp. 90–103.

Ni S, Tseng Y, Chen Y, Sheu J. "The broadcast storm problem in a mobile Ad Hoc Networks". Proceedings of ACM/IEEE MOBICOM'99; 1999. pp. 151–162.

Olariu S, Simplot-Ryl D, Stojmenovic I. "Localized communication and topology protocols for Ad Hoc networks: a preface to the special section". IEEE Trans Parallel Distrib Syst 2006;17(4):289–291.

Onat FA, Stojmenovic I, Yanikomeroglu H. "Generating random graphs for the simulation of wireless Ad Hoc, actuator, sensor, and internet networks". Pervasive and mobile computing: Elsevier, to appear.

Polastre J, Hill J, Culler D. "Versatile low power media access for wireless sensor networks". Proceedings of ACM Sensys; 2004.

Rhee I, Warrier A, Aia M, Min J. "Z-MAC: a Hybrid MAC for wireless sensor networks". Proceedings of ACM Sensys; 2005.

Rodoplu V, Meng TH. "Minimum energy mobile wireless networks". IEEE J Sel Araes Commun 1999;17(8):1333–1344.

Roundy S, Frechette LG. "Energy scavenging and non-traditional power sources for wireless sensor networks". In: Stojmenovic I, editor. Handbook of sensor networks: Wiley; 2005.

Ruiz-Ibarra E, Villasenor-Gonzalez L. "Cooperation mechanism taxonomy for wireless sensor and actor networks". IFIP Conference on Wireless Sensor and Actor Networks; Ottawa; 2008 Jul 14–15.

Steel D. "Smart dust". ISRC Technol Brief; 2005.

Stojmenovic I. Handbook of sensor networks: algorithms and applications: Wiley; 2005.

Stojmenovic I. "Energy conservation in sensor and sensor-actuator networks". In: Wu S-L, Tseng Y-C, editors. Wireless Ad hoc networking: personal-area, local-area, and sensory-area networks: Auerbach Publications, T&F; 2007. pp. 107–133. Chapter. 4.

Stojmenovic I, Lin X. "Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks". IEEE Trans Parallel Distrib Syst 2001a;12(10):1023–1032.

Stojmenovic I, Lin X. "Power-aware localized routing in wireless networks". IEEE Trans Parallel Distrib Syst 2001b;12(10):1–12.

Stojmenovic I, Nayak A, Kuruvila J. “Design guideline for routing protocols in Ad Hoc and sensor networks with a realistic physical layer”. IEEE Commun Mag 2005a;43(3):101–106.

Stojmenovic I, Nayak A, Kuruvila J, Ovalle-Martinez F, Villanueva-Pena E. “Physical layer impact on the design and performance of routing and broadcasting protocols in Ad Hoc and sensor networks”. Comput Commun 2005b;28(10):1138–1151.

Tanenbaum AS, Gamage C, Crispo B. “Taking sensor networks from the lab to the jungle”. IEEE Comput 2006:98–100.

Thrun S, Diel M, Hahnel D. “Scan alignment and 3-D surface modeling with a helicopter platform”. Proceedings of International Conference on Field and Service Robotic (FSR’03); 2003.

Toh CK. “Maximum battery life routing to support ubiquitous mobile computing in wireless Ad Hoc networks”. IEEE Commun Mag 2001;39(6).

Tso FP, Zhang L, Jia W. “Video surveillance patrol robot system in 3G, internet and sensor networks”. Proceedings of SenSys’07; 2007. pp. 395–396.

Tully A. “Pervasive tagging, sensors, and data collection”. Foresight Intelligent Infrastructure Systems Project; 2005.

Wark T, Crossman C, Hu W, Guo Y, Valencia P, Sikka P, Corke PI, Lee C, Henshall J, Prayaga K, O’Grady J, Reed M, Fisher A. “The design and evaluation of a mobile sensor/actuator network for autonomous animal control”. Proceedings of 6th International Conference on Information Processing in Sensor Networks (IPSN 2007); Cambridge; 2007. pp. 206–215.

Xia F, Tian YC, Li Y, Sun Y. “Wireless sensor/actuator network design for mobile control applications”. Sensors 2007;7:2157–2173.

Ye W, Heidemann J, Estrin D. “Medium access control with coordinated adaptive sleeping for WSNs”. IEEE/ACM Trans Network 2004;12(3):493–506.

ZigBee Alliance. “Network specification”, Version 1.0; 2004.

Chapter 2

Energy-Efficient Backbones and Broadcasting in Sensor and Actuator Networks

Hai Liu[1], **Amiya Nayak**[2], **and Ivan Stojmenovic**[2]

[1]*Hong Kong Baptist University, Hong Kong, P.R. China*
[2]*School of Information Technology Engineering, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5*

Abstract

This chapter first discusses backbones as subsets of sensors or actuators that suffice for performing basic data communication operations. They are applied in this chapter for energy-efficient broadcasting. In a broadcasting (also known as *data dissemination*) task, a message is to be sent from one node, which could be a sink or an actuator, to all the sensors or all the actuators in the network. The goal is to minimize the number of rebroadcasts while attempting to deliver messages to all sensors or actuators. Neighbor detection and route discovery algorithms that consider a realistic physical layer are described. An adaptive broadcasting protocol without parameters, suitable for delay tolerant-networks, is further discussed. We also survey existing solutions for minimal energy broadcasting problems, where nodes can adjust their transmission powers.

2.1 BACKBONES

A backbone is a subset of nodes that are able to perform assigned tasks and serve nodes which are not in the backbone. Thus, the backbone construction depends on the task to be carried. This chapter deals with backbones for the communication between nodes. The next chapter will discuss backbones for sensor area coverage.

The exact definition depends on the tasks or the particular desirable properties of the backbone. In wireless sensor networks, a backbone could be the set of active sensors while the rest of the sensors are sleeping (the problem of deciding which sensors should be active is often called the *activity scheduling* problem). The backbone of a network is normally required to be connected, such that the backbone nodes are able to communicate to perform assigned tasks. For instance, connected backbone nodes in *ad hoc* networks can perform efficient routing and broadcasting. Use of a backbone in wireless sensor networks could not only prolong the network lifetime, but could also make the operation of the network efficient. For instance, typical broadcasting in sensor networks is normally flooding-based, where each node retransmits the broadcasting message that it receives. Only nodes in the connected backbone retransmit the message in backbone broadcasting. Broadcasting via backbone could avoid a lot of useless retransmissions especially for dense networks. Backbones are often used to improve the routing procedure. A source node forwards the message to a backbone node. Routing then proceeds among backbone nodes until a backbone node is forwarded to the destination node.

There are basically three well-known methods to construct backbones: *grid partitioning-based* backbone (Xu *et al.*, 2001), *clustering-based* backbone (Lin and Gerla, 1997), and *connected dominating set* (*CDS*)*-based* backbone (Guha and Khuller, 1998). In the grid partitioning-based backbone, the area of the network is divided into grids and one node in each grid is selected as a backbone node. The size of grid should be carefully determined to guarantee that the backbone is connected. In clustering-based backbone, nodes are grouped into clusters and a node is elected as the clusterhead (CH) in each cluster. Any node in the network is either a CH or a neighbor of a CH. Additional nodes are required to be included to make the CHs connected. The concept of CDS has been introduced in Chapter 1, and will be elaborated on again later in this chapter.

The *maximal independent set* (MIS) is an important concept which is used in the construction of some clustering and CDS-based backbones. An independent set is a set of vertices in a graph where no two vertices are adjacent. A MIS is an independent set which is not a subset of any other independent set. The largest MIS is called the *maximum independent set*.

The independent set problem is equivalent to the clique problem since an independent set in a graph G is a clique in the complement graph of G. Therefore, finding the maximum independent set is equivalent to finding the maximum clique which is a well-known NP-complete problem. However, a MIS can be found in polynomial time by adding vertices one by one until no more vertices could be added.

Backbone construction algorithms are normally applied to unit disk graph (UDG) where all nodes are assumed to have a common transmission range. In this chapter, we first separately consider sensor and actuator networks since each of them can be modeled as UDG. Later in Section 2.8, we discuss joint backbones for sensor and actuator networks. Backbone concepts, where nodes have an adjustable transmission range are studied in literature but are not covered here because of perceived lack of their usefulness. Therefore, most sections here assume that nodes have fixed transmission radius (exceptions are Sections 2.10 and 2.8 discussing heterogeneous networks of both sensors and actuators).

2.2 GRID PARTITIONING-BASED BACKBONES

A grid partitioning algorithm for backbone construction, called *geographical adaptive fidelity* (*GAF*), was proposed by Xu *et al.* (2001). It assumes that location information is available via GPS, and each node knows its current location relative to other nodes. The algorithm divides the whole area of the network into virtual grids. The virtual grid is defined such that, for any two adjacent grids, any node in one grid can directly communicate with any node in the other grid. That is, all nodes in the same grid are "equivalent" from the routing or broadcasting point of view. Therefore, one representative node from each grid is sufficient to construct a connected backbone.

Suppose r is the size (edge length) of the virtual grid and R is the transmission range. In order to guarantee that any two nodes in adjacent grids can communicate with each other, the following relationship is required: $r^2 + (2r)^2 \leq R^2$. Thus, we have $r \leq R/\sqrt{5}$.

Geographical adaptive fidelity was further studied by Basagni *et al.* (2004). This study shows that the backbone constructed by GAF may disconnect the graph. The reason is that nodes are not evenly distributed in the grids. It is very likely for a grid with one or more nodes to be adjacent to one or more empty grids. Two nonadjacent nonempty grids may be connected when all the nodes within them are active. However, particular leaders computed by GAF may not be connected and this results in a partition of the backbone. The partition of the backbone is illustrated in Figure 2.1 (not all edges of UDG are drawn). Other disadvantages include the use of parameter (grid size) and global synchronization for grid boundaries.

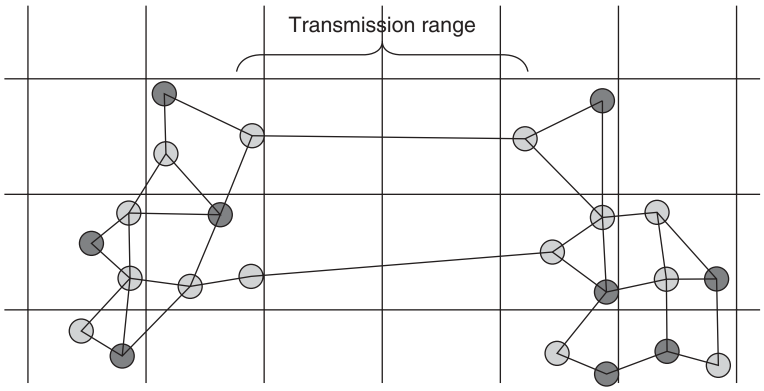


Figure 2.1 Partition of the backbone by using GAF.

2.3 CLUSTERING-BASED BACKBONES

Clustering is one of the commonly used backbones to organize the network in a hierarchical architecture. It is used to partition nodes of the network into groups (clusters) in which CHs dominate the other nodes in the clusters. Clustering provides spatial reuse of the bandwidth which is a limited resource in wireless sensor networks. Moreover, clustering provides a hierarchical architecture for efficient routing. Existing solutions for clustering usually consists of two phases: clustering construction and clustering maintenance. In the first phase, nodes are chosen to act as coordinators of the clusters (CHs). A CH and some of its neighbors form a cluster. After clustering construction, clustering maintenance is required to reorganize the clusters due to node mobility and node failure.

There are different cluster structures. Clusterheads may or may not be allowed to be neighbors, and other nodes may or may not be always connected to a CH. For example, nodes in LEACH (low-energy adaptive clustering hierarchy) protocol proposed by Heinzelman *et al.* (2000) randomly decide whether or not to become CHs. The parameter used in decision making is the percentage of desired CHs in the network. Sensors that decide to become CHs broadcast their decision. Each node reports to the CH with the highest signal strength, and thus clusters correspond to Voronoi diagrams of CHs. A CH allocates a timeslot for each of its cluster members for reporting aggregation data. Selection of CHs is periodically repeated to balance energy consumption of nodes. The structure of the backbone computed by LEACH protocol is illustrated in Figure 2.2. The structure of the clusters constructed by LEACH is inefficient since the sink may be very far from many CHs. Consequently, direct reporting may be extremely energy-consuming or even impossible. Furthermore, two CHs may be neighbors of each other, and many nodes may not have any CH as a direct neighbor. There are dozens of recent articles describing multihop reporting or better CH decisions, compared to LEACH. In particular, Xia and Vlajic (2006) proved that only clustering schemes that position their resultant clusters within the isoclusters of the monitored phenomenon are guaranteed to reduce the nodes' energy consumption and extend the network lifetime. They also proposed the first clustering algorithm that employs the similarity of the nodes' readings as the main criterion in cluster formation.

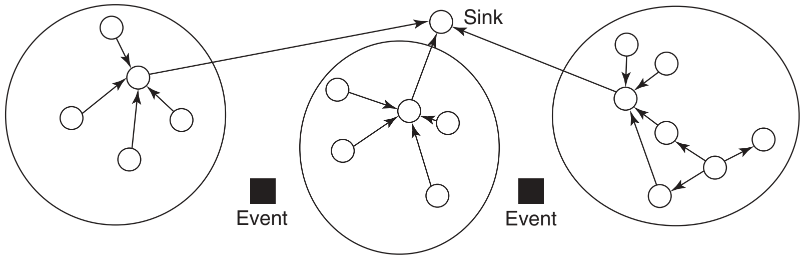


Figure 2.2 LEACH protocol.

A class of clustering algorithms requests that no two CHs could be direct neighbors, and any other node should be adjacent to at least one CH. A representative is the algorithm by Lin and Gerla (1997), and its variants. It assumes that each node has a unique node *key* and knows the *key*s of its one-hop neighbors. The basic idea of the CH algorithm is to use the node *key* as a priority indicator when selecting CH in each cluster. It works as follows. Each node compares its *key* with the *key*s of its neighbors. Initially all nodes are undecided. If an undecided node has the lowest *key* among its undecided neighbors then the node decides to create its own cluster and broadcasts the decision and its *key* as the cluster *key*. Upon receiving a message from a neighbor that announces itself to be a CH, each undecided node will declare itself as a non-CH (and decided) node and will inform its neighbors by transmitting a message. The declaration of non-CH will encourage more CHs to create clusters.

In the example in Figure 2.3, IDs of nodes (numbers from 1 to 20) are used as node *key*s, with their natural ordering ($1 < 2 < \ldots < 20$). In the first round, nodes 1, 5, 10, and 18 have the lowest *key*s among their neighbors and become CH. The CHs in the first round send their decisions to their neighbors, which then declare themselves non-CHs and inform their neighbors. In the second round, nodes 3 and 15 create their clusters since they receive a non-CH announcement from nodes 2 and 14, respectively. Nodes are therefore divided into 6 clusters: {1, 2}, {3, 2, 4, 11}, {10, 11, 12, 13, 14}, {15, 14, 16, 17}, {5, 6, 7, 8, 9, 16}, {18, 19, 20}. Any non-CH node that receives more than two CH declarations will declare itself as a gateway node, and will belong to all corresponding clusters (alternatively, it can be assigned to only one of these clusters by some criteria). In the example, nodes 2, 6, 11, 14, 16, 20 are gateway nodes which connect CHs. Any two neighboring CHs are two or three hops away from each other. They share a gateway node if they are at a distance of two hops (e.g., path 1-2-3). Some paths between two CHs at a distance of three hops may contain nodes that belong to a single cluster. For example, path 18-20-6-5 contains one such as the node 20. These nodes are also considered as gateway nodes, since otherwise the network of CHs and gateway nodes might be disconnected.

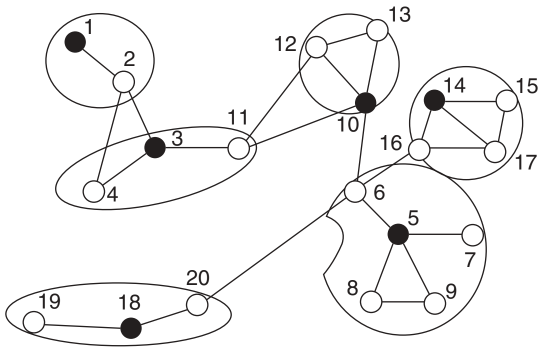


Figure 2.3 Clusterheads 1, 5, 10, 18 in round 1 and 3 and 15 in round 2.

A distributed clustering algorithm, called distributed mobility-adaptive clustering (DMAC), proposed by Basagni (1999) uses a mechanism to construct clusters which is similar to the algorithm in Lin and Gerla (1997). However, it uses the weight of the nodes instead of node IDs as keys when determining the CHs. The weight of the nodes could be assigned depending on the remaining energy in the cluster or the capacity of the nodes. The algorithm (Lin and Gerla, 1997) is then followed with such weight instead of the original lowest ID used in Lin and Gerla (1997). On the basis of the DMAC, a protocol for the topology control of large wireless sensor networks, called *S-DMAC*, was proposed in Basagni *et al.* (2004). The protocol is used to select a subset of sensor nodes to build a connected backbone and let all other nodes switch to an energy-conserving "sleep mode." A connected backbone consists of backbone nodes and gateway nodes which interconnect the backbone nodes, where backbone nodes are the CHs computed by DMAC. S-DMAC optimizes the overhead for neighbor discovery at both the backbone construction and backbone maintenance stages by limiting the use of "hello" messages. The backbone is reorganized only when introducing a new batch of nodes with much higher energy than the current nodes, or when backbone nodes deplete their energy. A nonbackbone node will join a newly inserted backbone node only when the residual energy of the new backbone node exceeds the original one's energy by a predefined threshold. When compared with GAF, S-DMAC provides a connected backbone with a smaller number of nodes. It also reduces the number of backbone reorganizations and exchanged "hello" messages.

Clustering is a typical example of a quasi-local protocol for both the construction and maintenance phases (Simplot-Ryl *et al.*, 2005). It has a "chain effect": a simple change in an edge or node failure/addition may trigger global backbone updates by propagation. The propagation is typically suppressed at the expense of quality of the cluster structure. Another difficulty is to connect CHs into a connected backbone via gateway nodes. The process normally either elects too many gateways, or uses too many messages to elect fewer ones.

2.4 CONNECTED DOMINATING SETS AS BACKBONES

Connected dominating sets are one of the primary techniques used to build backbones for wireless sensor and *ad hoc* networks. The concept of CDS can be briefly introduced as follows. A dominating set (DS) is a subset of the vertices of a graph where every vertex in the graph is either in the subset or is adjacent to at least one vertex in the subset. A CDS is a connected DS. The nodes in CDS are called *dominators*, while other nodes are called *dominatees*. There are several metrics used to evaluate the quality of a backbone protocol. They evaluate its properties such as backbone size, protocol duration, message overhead and backbone robustness (Basagni *et al.*, 2006). It is normally desirable to construct a backbone with the smallest size possible and thus prolong the lifetime of the network and alleviate congestion. That is, it is useful to find the minimum connected dominating set (MCDS). However, finding the MCDS in general graphs has been shown to be equivalent to the set cover problem which is NP-hard (Garey and Johnson, 1978). The MCDS remains NP-hard even in UDGs (Clark *et al.*, 1990). Therefore, approximation algorithms and heuristic algorithms have been proposed in literature to solve the problem. Approximation algorithms are normally evaluated by the *approximation/performance ratio*, which is upper bound on the ratio of the algorithm's performance to the optimal algorithm's performance. Algorithms introduced next can be classified into centralized algorithms (e.g., Guha and Khuller, 1998) and localized algorithms (e.g., Adjih *et al.*, 2005). Some distributed algorithms as implementation of centralized algorithms are also introduced (e.g., Alzoubi *et al.*, 2002).

2.4.1 Centralized Set Cover-Based Algorithms

Guha and Khuller (1998) proposed centralized algorithms to build a CDS for general graphs. The basic idea of the first algorithm is to construct the CDS by growing it from a single node outward. The second algorithm, on the other hand, constructs CDS simultaneously from several nodes. The first algorithm runs as follows. Each node is initially colored white. Next, the node with the largest degree is colored black and all its neighbors are colored gray. This last step is repeated until there are no white nodes left in the graph. Each time, the gray node with the largest number of white neighbors is colored black and then all its white neighbors are colored gray. Node IDs can be used to break ties. Finally, all black nodes form a CDS. The algorithm is illustrated in Figure 2.4. Nodes 3 and node 5 have the largest degree. Since $3 < 5$, node 3 is colored black in

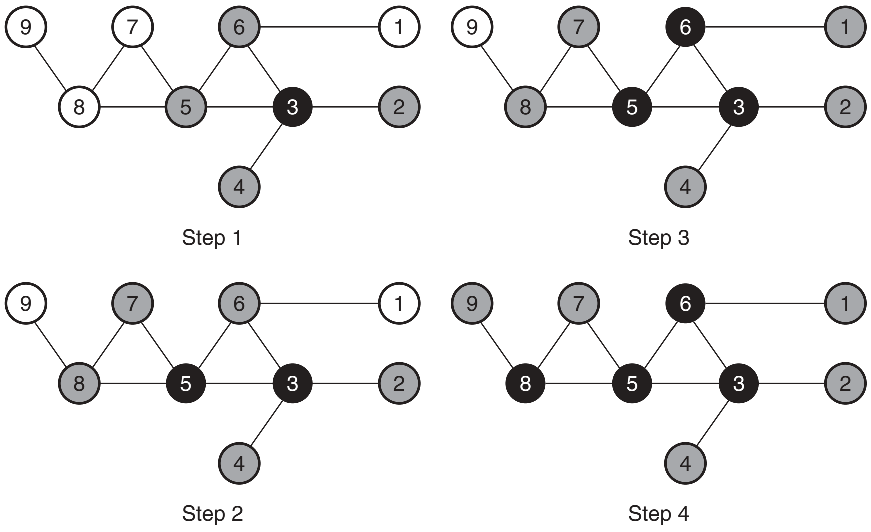


Figure 2.4 Guha–Khuller algorithm for CDS construction.

the first step. All its neighbors are colored gray. Since node 5 has the largest number of white neighbors, it is colored black next and all neighbors of node 5 are colored gray. Similarly, node 6 is colored black and node 1 is colored gray in step 3. After node 8 is colored black, all black nodes {3, 5, 6, 8} form a CDS.

Cheng *et al.* (2003) showed that there exists a polynomial-time approximation scheme (PTAS) for MCDS in UDGs. This means that the performance ratio for polynomial-time approximations can be $1 + \varepsilon$ for any given $\varepsilon > 0$. However, the running time grows rapidly as ε approaches 0. Thus, PTAS is not applicable in practice. Min *et al.* (2006) proposed to use a Steiner tree with a minimum number of Steiner nodes (ST-MSN) to construct a MIS.

The algorithm (Min *et al.*, 2006) consists of two phases. In the first phase, a MIS is constructed (Wan *et al.*, 2004), such that every subset of the MIS is two hops away from its complement. Suppose I is the MIS and A is an arbitrary subset of I. The distance of A and $I - A$ is exactly two hops. That is, there must exist $a \in \mathrm{I}$ and $b \in \mathrm{I} - A$, such that a is two hops away from b. Note that any MIS is also a DS. All the nodes in the MIS are colored black and all other nodes are colored gray. In the second phase, a gray node that is adjacent to at least three connected black components is colored black. If no node satisfies this condition, a gray node that is adjacent to at least two connected black components is selected to be colored black. Finally, all black nodes form a CDS. The approximation ratio of the algorithm has been proved to be 6.8 for UDGs. It means that the number of nodes in CDS is at most 6.8 times of that in MCDS in UDGs.

2.4.2 MIS-Based CDS

The basic idea of the algorithms in this category is to compute and then connect an MIS. There are basically two strategies to compute a MIS. One is based on a single leader and the other is based on multiple leaders. Two representative algorithms for these strategies are introduced next.

Alzoubi *et al.* (2002) proposed two versions of a single leader-based algorithm for CDS construction. In both algorithms, the distributed leader election algorithm (Cidon and Mokryn, 1998) is employed to construct a rooted spanning tree. The basic idea is to partition all nodes in the network into several groups. Each group contains a candidate node and its domain of supportive nodes. These groups are iteratively merged until there is only one candidate which eventually becomes the root (leader) of the tree. Then, an iterative labeling strategy is used to classify the nodes in the tree to be either black (dominator) or gray (dominatee) based on their ranks. The rank of a node is a pair {hop_count, ID}, where hop_count is the number of hops to the root of the spanning tree. The labeling process starts with the root and finishes at the leaves. At the initialization stage, the node with the lowest rank colors itself black and broadcasts a "dominator" message. If a node receives any "dominator" message, it colors itself gray and broadcasts a "dominatee" message. If a node receives "dominatee" messages from all its neighbors with lower rank then it will color itself black and send a "dominator" message. The algorithm terminates when the leaf nodes are reached. All black nodes form a MIS.

To connect the nodes in the MIS to form a CDS, initially, the root joins the CDS and broadcasts an "invite" message. The "invite" message is relayed to all two-hop neighbors from the current CDS. Upon receiving the "invite" message for the first time, a black node joins the CDS to the gray node which relayed the message (and that gray node joins CDS). In turn, the black node broadcasts an "invite" message. The process terminates when all the black nodes have joined the CDS. The performance ratio of the resulting CDS has been shown to be at most 8.

The algorithm is illustrated in Figure 2.5. Suppose the leader election algorithm (Cidon and Mokryn, 1998) is used to compute a spanning tree which is rooted at node 0. The solid lines represent the edges in the spanning tree while the dashed lines represent the edges in the UDG (each edge of the spanning tree also belongs to the UDG). According to the algorithm, node 0 is colored black and sends a "dominator" message to its neighbors 2, 4 and 12 (Fig. 2.5a). Upon receiving the message, nodes 2, 4 and 12 are colored gray and send "dominatee" messages. For example, node 4 sends the "dominatee" messages to its neighbors 0, 5, 7, and 10. Node 5 is now colored black since it has received "dominatee" messages from all its neighbors with lower rank (node 4 only). Figure 2.5b shows the colors of the nodes after the labeling process. The final step is to build the dominating tree from the root. Node 0 broadcasts an "invite" message to its neighbors which relay the message to its two-hop black neighbors 3, 5, and 7. These black nodes join the dominating tree with their gray neighbors, 2 and 4, which relay the message. Finally, all of the black nodes in the dominating tree form a CDS (Fig. 2.5c).

Cheng *et al.* (2004) proposed a multiple leader-based algorithm for CDS construction. The basic idea of the algorithm is as follows. Initially, each node with the smallest ID among its one-hop neighbors marks itself as a leader. The leaders serve as roots of trees which form a forest. Then, chains of one or two nodes are computed to connect neighboring trees. Messages exchanged during the algorithm are linear in the number of nodes in the network and the performance ratio was shown to be at most at 147.

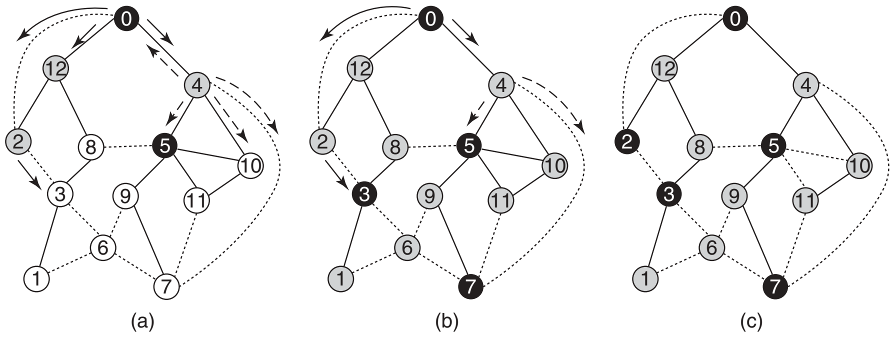


Figure 2.5 MIS-based algorithms.

The drawbacks of MIS-based CDS algorithms in this section are the use of significant number of messages to construct and maintain the structures. In realistic scenarios, with medium access and realistic physical layers, these messages may lead to errors in decisions, in addition to causing message overhead and energy consumption.

2.4.3 Coverage-Based CDS

Wu and Li (1999) proposed a localized algorithm to construct CDS in general graphs. It assumes that each node knows the local topology of its two-hop neighbors. That is, it has a list of its direct neighbors, and lists of direct neighbors of each if its neighbors (obtained by exchanging two rounds of "hello" messages). Alternatively, each node should be aware of its own position, and the position of all its neighbors, which requires one round of "hello" messages, where each node informs its neighbors about its own position. We describe a modified definition by Stojmenovic *et al.* (2002) (followed by a correction and comments in Stojmenovic (2004)) since it completely eliminates the need for sending any additional message (in addition to mentioned "hello" messages for gathering needed local knowledge). More precisely, each node can decide for itself whether or not it is in CDS without any message exchange with neighbors. However, to learn the CDS status of neighboring nodes, communication may be required. The two definitions and construction algorithms (Wu and Li, 1999) and (Stojmenovic *et al.*, 2002) always (for any input graph) produce the same CDS. We also assume that each node has a unique *key*, and that these keys are distributed to neighbors as part of required local knowledge.

A node is a *0-gateway* node if it has two unconnected neighbors. A node, say A, is "covered" by a neighbor, say B, if each neighbor of A is also a neighbor of B and $key(A)$ is less than $key(B)$. A node A is covered by two connected neighboring nodes B and C if each neighbor of A is also a neighbor of at least one of nodes B or C, $key(A)$ less than $key(B)$ and $key(A)$ less than $key(C)$. A 0-gateway node not covered by any neighbor becomes a *1-gateway* node. A 1-gateway node not covered by any two connected neighbors becomes a *2-gateway* node. Finally, the 2-gateway nodes form a CDS. Note 0-gateway and 1-gateway nodes also create CDS, since 2-gateway nodes are their subsets. In the example in Figure 2.6, nodes 6, 8, 14, 15, 16, 17 are not 0-gateway nodes. Node 5 is covered by its neighbor 9 since all 5's neighbors are also neighbors of 9 and 5 less than 9 (keys are assumed to be ordered numerically, $1 < 2 < \cdots$). Thus, 0-gateway node 5 is not a 1-gateway node. Node 2 is covered by two connected neighbors 3, 12 since they have higher key values and the remaining neighbors 4, 6 and 16 are covered by 3 and 12. Therefore, 1-gateway node 2 is not a 2-gateway node. Finally, the remaining 2-gateway nodes {1, 3, 4, 7, 9, 10, 11, 12, 13} form a CDS.

The key of a node could be its ID (Wu and Li, 1999), a pair (*degree, ID*) (Stojmenovic *et al.*, 2002) (degree is the number of neighbors of a node) or a three-tuple (*energy, degree, ID*) (Wu *et al.*, 2002, 2003). For example, when

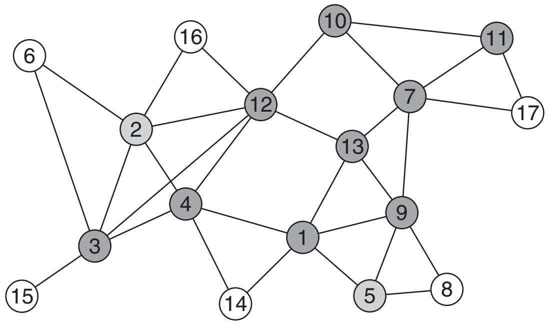


Figure 2.6 CDS by coverage-based algorithms.

(energy, degree, ID) is used, a node with a higher energy level has a higher probability to be a dominator. If the energy levels of two nodes are equal, then the node degree is used to break the tie. Similarly, if both energy level and node degree are the same, node ID is used to break the tie. Such choice for a key will balance the energy of nodes since generally nodes with higher energy are selected to be active. The decisions can be periodically reevaluated, so that the network lifetime is prolonged. Shaikh *et al.* (2003) proposed several alternative key definitions, based on combinations of node degrees and remaining energy levels. A surprising winner was $key = energy/degree + energy$, but no explanation was given for such an outcome.

Size of the CDS in Figure 2.6 could be further reduced by applying a generalized rule (Dai and Wu, 2003), where coverage could be provided by an arbitrary number of connected one-hop neighbors. In the example in Figure 2.6, node 7 is covered by four connected neighbors 9, 10, 11, 13 since the remaining neighbor 17 is covered by 11 and the key value of 7 is the minimal. Thus, node 7 can be removed from the CDS. Formally, the rule can be defined as follows. Let *N(u)* be the set of *u*'s connected neighbors which have higher key values than *A*. If any of *u*'s neighbors is either in *N(u)* or is a neighbor of a node from *N(u)*, then node *u* can be removed from the CDS. This rule formulation has been proposed by Stojmenovic *et al.* (2004) in order to avoid similar message exchanges among neighbors. Nodes that remain in CDS by this definition will be called *gateway* nodes here.

A simple algorithm for verifying this condition is proposed by Carle and Simplot-Ryl (2004). First, each node checks if it is 0-gateway node. Then, each 0-gateway node u constructs a subgraph G_h (of original UDG) which consists of its one-hop neighbors with higher key values and existing edges between them. If G_h is empty or disconnected, u decides to join the DS. If G_h is connected but there is a neighbor of u which is not a neighbor of any node in G_h, u also decides to join the DS. Otherwise, u is covered and is not in the DS. Dijkstra's shortest path algorithm could be used locally at each node to test the connectivity. Note that the required connectivity of neighbors does not imply that any two of them are the one-hop neighbors, but only that the subgraph of these nodes is a connected graph.

We will refer to nodes selected into DS by the generalized rule as *gateway* nodes (generalizing the previous definition). These are nodes not covered by any subset of their connected neighbors. Thus, each node which is not selected as a gateway node to join CDS is either not a 0-gateway node or there exists a set of k connected neighbors with higher key values that covers it. It is not immediately clear whether or not the so defined gateway nodes create a (connected) DS. Each gateway node is also a k-gateway node for any k. We will now give a proof of the CDS property. The proof of connectivity is similar to the one given in Wu and Li (1999) while the proof of DS property is generalized from the one given in Stojmenovic (2004).

Theorem 2.1. Gateway nodes create CDSs.

Proof. Suppose that, on the contrary, the created set S of gateway nodes is not a DS. Then, there exist some nodes which are not in S, and which have no neighbors among nodes in S. Among such nodes, let X be the one with the largest *key* value. If all neighbors of X are not 0-gateway nodes, the graph is a complete graph. Otherwise, let Y be the 0-gateway neighbor of X with the largest *key*. Since Y is not a gateway node, it is covered by k of its connected neighbors $U_1, U_2, \ldots, U_k$ (this is not necessarily a chain of these neighbors), and has the lowest key among them. Such neighbors that are not 0-gateway nodes can be eliminated first without affecting coverage property. X then must be a neighbor of at least one of these (remaining) 0-gateway nodes, say U_i. However, $key(Y) < key(U_i)$ contradicts the choice of Y. Therefore, the set of nodes, not in S and not a neighbor of any node in S, is empty, and therefore S is a DS.

The connectivity of created DS follows from the observation that removing any node, which decides not to participate in CDS, does not disconnect any existing path between a node S and node D (Wu and Li, 1999). In the example in Figure 2.7, suppose that S was connected to D by a path $S \rightarrow \ldots \rightarrow A \rightarrow U \rightarrow B \rightarrow \ldots \rightarrow D$, which is composed of bold edges. Suppose node U is not in DS. That is, U is covered by k ≥ 1, of its connected neighbors $U_1, U_2, \ldots, U_k$. In Figure 2.7, $k = 3$. Thus, nodes A and B are connected via $U_1, U_2, \ldots, U_k$. The path segment $A \rightarrow U \rightarrow B$ can be replaced

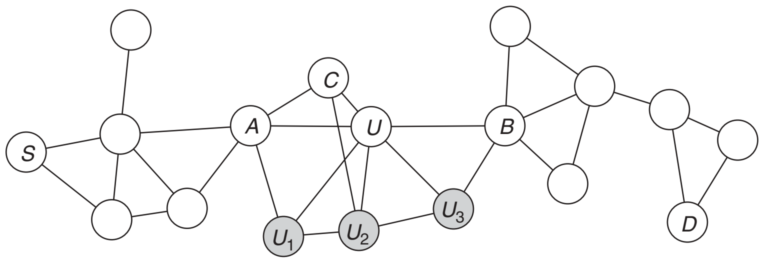


Figure 2.7 Connectivity of gateway nodes.

with $A \rightarrow U_i \ldots \rightarrow B$. In Figure 2.7, $A \rightarrow U \rightarrow B$ can be replaced with $A \rightarrow U_1 \rightarrow U_2 \rightarrow U_3 \rightarrow B$. Therefore, U can be removed while connectivity is still preserved. All nodes can be sorted by their keys, and parallel removal can be considered as a series of removals of nodes, with increased key values. The connectivity is then preserved at each step, as nodes previously removed are not used in the removal of new nodes (each node is covered only by higher key neighbors). ♦

Enhanced DSs are further studied by Dai and Wu (2003) and Ingelrest *et al.* (2007). An enhanced definition for computing a DS is presented here. An intermediate node u is not a dominator if there exists in its two-hop neighborhood a connected set A_u of nodes with higher key value, such that each neighbor of u either belongs to A_u or is a neighbor of a node in A_u. In the example in Figure 2.8a, after applying the algorithm (Wu and Li, 1999), all black nodes form a CDS. However, its size can be reduced as in Figure 2.8b. Since a is covered by the two-hop neighbors $\{b, e, f\}$, a is removed from CDS. Similarly, b and all of its neighbors $\{a, c, e\}$ are covered by its connected higher ID two-hop neighbors $\{e, f\}$. Thus, b is removed from CDS.

The performance ratio of the algorithm (Wu and Li, 1999) has been shown to be $O(n)$, where n is the number of nodes in the network (Wan *et al.*, 2004). An example of a worst case behavior are nodes at square grid points (x, y) where (x, y) is used also as a key value [with a large transmission, but not too large, for example, $n^{0.5}/2)$]. All internal points of this mesh remain in CDS. A simpler example of a worst case behavior is a chain of nodes with increased key values (and transmission radius that gives e.g., about $n/2$ neighbors per node), which forces almost all nodes to be in CDS. However, Dai and Wu (2004) proved that the generalized CDS concept has a constant approximation ratio on average, and a very low probability of having an infinitely large approximation ratio. Wiese and Kranakis (to appear) proved that no algorithm, based on the one-hop positional information, can achieve constant approximation ratio in the worst case. The proof is based on an example of an arbitrary number of nodes placed on two concentric circles, with distance equal to the transmission radius R, so that nodes are paired, one on each circle, with a distance R between paired nodes. If uv is one such pair

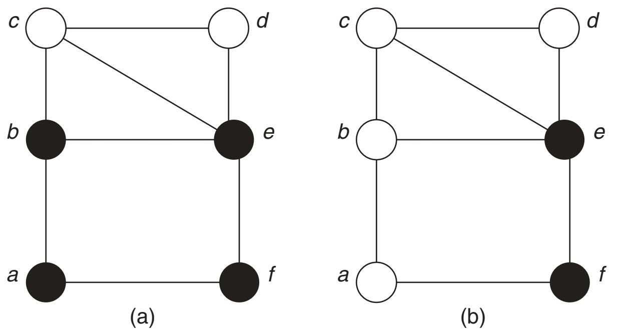


Figure 2.8 (a) CDS by generalized rule (b) Enhanced CDS.

then, based on local knowledge of positions of neighbors, both u and v decide to join CDS since the other node appears disconnected from the rest in the local one-hop subgraph. However, a CDS with a constant number of points on each circle exists, thus showing that the worst case approximation ratio is unbounded.

Major advantages of the algorithm are zero communication overhead (after "hello" messages to learn neighbors) and its locality of maintenance. In case of movement or on/off switching of a node, the algorithm only requires neighbors of the node to update their status. The performance of the algorithm (Wu and Li, 1999) (more precisely, the variant described here, from (Stojmenovic *et al.*, 2002) was used) was studied in Basagni *et al.* (2006) via extensive simulation. The results show that it is a very fast algorithm and achieves the best trade-off balance between the low message complexity and size of CDS, compared to all alternatives (most of them are described here).

2.4.4 Topology Control in 802.15.4-Based Sensor Networks

Topology control for star-shaped topology of IEEE 802.15.4 (IEEE Standard 802, 2003) based sensor networks (see also Chapter 1 for more detail) was studied by Ma *et al.* (2007). This topology defines two roles for a node: *coordinator* or *device*. A coordinator serves its associated devices by buffering and routing packets for them, such that the devices can go to sleep mode and wake up only when they need to communicate with the coordinator. Authors Ma *et al.*, (2007) prove that a route between two nodes is bidirectionally connected if and only if all intermediate nodes along the route are coordinators. This is because a device (noncoordinator) node on the route cannot be synchronized at the same time with the previous and subsequent node on the route. This means that the CDS concept of backbone is implied by the medium access protocol for star-shaped topology of IEEE 802.15.4. The duty cycle of every coordinator node is therefore 100%, while device nodes may periodically sleep. It is therefore important to select small size CDSs as coordinators.

Three localized pruning algorithms were proposed to construct CDS (Ma *et al.*, 2007). The *self-pruning* algorithm works as follows. Let $S(v)$ be the set of neighbors of u which have a higher priority than v. Node v is not a coordinator if (i) $S(v)$ is nonempty and connected; (ii) any neighbor of v is covered by $S(v)$ [i.e., it is either in $S(v)$ or is a neighbor of node from $S(v)$]. The priority of a node can be based on its energy level, node ID or node degree. Note that this definition is the same as generalized rule CDS concept of (Dai and Wu, 2003; Carle and Simplot-Ryl, 2004; Stojmenovic *et al.*, 2004).

Ordinal pruning (Ma *et al.*, 2007) is a similar concept. The only difference is that $S(v)$ includes not only neighbors with higher priorities but also neighbors with lower priorities that have already become coordinators. Therefore the coordinator node set of the ordinal pruning algorithm is a subset of that of the self-pruning algorithm. This variant of CDS concept was also described in Dai and Wu (2004).

In *layered pruning* method, instead of waiting for all the neighbors with lower priorities, each node only considers all neighbors with higher hop distances. That is, the set $S(v)$ in the definition contains all neighbors with higher priorities and all neighbors u with $d(u)$ greater than $d(v)$, where $d(u)$ is hop count from the sink. Thus all nodes in the same layer decide their roles simultaneously. The coordinator set of the layered pruning algorithm has been proven to be a subset of that of the self-pruning algorithm.

2.4.5 Multipoint Relaying-Based CDS

Several backbone schemes are based on the concept of multipoint relays (MPRs) of a node S, defined as a minimal size subset of neighbors of a given node S that will "cover" all the two-hop neighbors of S. It requires that each node gathers the two-hop knowledge. Node A is covered by a node B if A is a direct (one-hop) neighbor of B. Relay points of S are the one-hop neighbors of S that cover all the two-hop neighbors of S. The objective is to minimize the number of relay points of S. The computation of a MPR set with minimal size is an NP-complete problem (Qayyum *et al*., 2002). Relay points are determined by applying Guha–Khuller algorithm for CDS construction, which is restricted to a two-hop neighborhood of a given node. It is similar to a heuristic algorithm, called *greedy set cover algorithm*, proposed in Lovasz (1975). This algorithm repeats the selecting node B that maximizes the number of neighbor nodes that are not yet covered. In the example in Figure 2.9a, gray nodes 2, 5 and 6 are relay points of node 0 and the size is minimal.

Adjih *et al*. (2005) proposed an MPR-based algorithm for CDS (MPR-CDS) backbone construction. Each node computes its MPR set by selecting a subset of the one-hop neighbors which cover all the two-hop neighbors. The node attaches the relay list to a "hello" message which is broadcasted to its neighbors. Upon receiving the "hello" message, an intermediate node decides to join the CDS if it has either the smallest ID in its neighborhood or if it is the MPR for the neighbor with the smallest ID. Wu (2003) improved the rule by eliminating the node that has the smallest ID among its neighbors, but without two unconnected neighbors. The construction of the MPR-CDS backbone requires the two-hop neighbor knowledge, plus a message containing the list of relay nodes of each node. This can be treated overall as CDS construction requiring three rounds of messages, plus another round if the CDS decisions are to be communicated to neighbors.

Consider the example in Figure 2.9b. Since a and b are the nodes with the smallest ID amongst their neighbors, they decide to belong to the CDS. Node a computes its MPR set $\{g, h\}$ and then attaches the list to its "hello" message. Upon receiving the "hello" message, nodes g and h decide to belong to the CDS as well. Similarly, node f decides to belong to the CDS since it is the MPRs of b. Finally, nodes $\{a, b, f, g, h\}$ form a CDS by the algorithm (Adjih *et al*., 2005). Using the improved algorithm (Wu, 2003), node b is not selected for CDS, and CDS set is $\{a, f, g, h\}$.

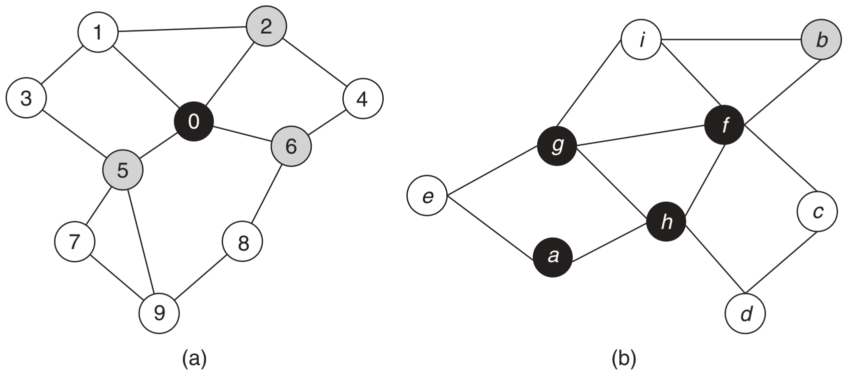


Figure 2.9 MPR-CDS algorithm. (a) Relay nodes 2, 5 and 6 of node 0; (b) MPR-CDS nodes $\{a, b, f, g, h\}$ by Adjih *et al*. (2005) and $\{a, f, g, h\}$ by Wu (2003).

2.5 OVERVIEW OF BROADCASTING TECHNIQUES

Broadcasting is one of the fundamental operations in wireless sensor and *ad hoc* networks. It is used to disseminate a message from a node, also called *a source*, to all other nodes in the network. In sensor and actuator networks, broadcasting usually comes from a sink or an actuator and it is directed to all of the sensors or all of the actuators in the network. Applications of broadcasting include paging a particular host or sending an alarm signal to all nodes in the network. Broadcasting in wireless sensor networks is normally used to disseminate request information for measurements such as temperature and noise level. Broadcasting is often referred to as *flooding* when it is employed in protocols to disseminate control messages such as route discovery, route maintenance, topology updating and synchronization. Since both sensor networks and actuator networks can be modeled by UDG, we first consider broadcasting separately in sensor networks or actuator networks. Then later we discuss sensor and actuator network as a single heterogeneous network.

Since the transmission range is restricted in wireless sensor actuator networks (WSANs) due to the limited energy of nodes, it is normally impossible for the source to directly cover all recipients in the network. Thus, many nodes have to act as routers by relaying the broadcasting message to their neighbors. The broadcasting message is propagated hop by hop and ultimately reaches all the nodes in the network. The simplest implementation of broadcasting is *blind flooding*. In blind flooding, every node in the network retransmits the flooding messages if it is its first time to receive the broadcasting message. If the network is connected and collisions are not considered, blind flooding guarantees that all nodes in the network can receive the message. However, since each node retransmits the flooded message, in dense networks many redundant packets are generated which may cause severe contention and collisions. This problem was studied by Ni *et al*. (1999) and was referred to as the *broadcast storm problem*.

It is illustrated in Figure 2.10. Suppose node *A* initiates a broadcast by broadcasting a message to nodes *B* and *C*. According to blind flooding, *B* and *C* retransmit the message if they have not been transmitted it before. Note that *B* and *C* have to contend for the broadcast medium in the shadow area as shown in Figure 2.10. Node *D* is inside the transmission coverage of both *B* and C. Therefore, *D* cannot correctly receive the flooding message if *B* and *C* retransmit the message at the same time. Moreover, both nodes *B* and *C* receive two copies of the message from *A* and each other. These redundant messages not only cost extra energy consumption but also increase the probability of collisions between the messages. In fact, either *B* or *C* rebroadcasting the flooding message is sufficient to cover node *D* in Figure 2.10.

To solve the broadcast storm problem, many broadcasting protocols have been proposed in literature. One of the primary techniques for efficient broadcasting is *backbone broadcasting* (Stojmenovic *et al.*, 2002). The broadcasting can be performed using any constructed backbone. Depending on the choice of algorithms, the set of retransmitting nodes may be a subset of backbone nodes, or may include additional nodes, gateways, to connect backbone ones (if not connected already).

Broadcasting protocols can be divided into two categories based on the way relay nodes are chosen: *sender-based* protocols and *receiver-based* protocols. In sensor-based protocols, each sender nominates a subset of its neighbors to be the next hop relay nodes. In receiver-based protocols, each receiver of a flooding message makes its own decision on whether or not it should forward the message. For example, the neighbor elimination scheme (Peng and Lu, 2000; Stojmenovic and Seddigh, 2000) and area-based beaconless broadcasting algorithm (ABBA) (Ovalle-Martinez *et al.*, 2006) are receiver-based while MPR protocol (Qayyum *et al.*, 2002) and efficient flooding protocol (Liu *et al.*, 2007) are sender-based.

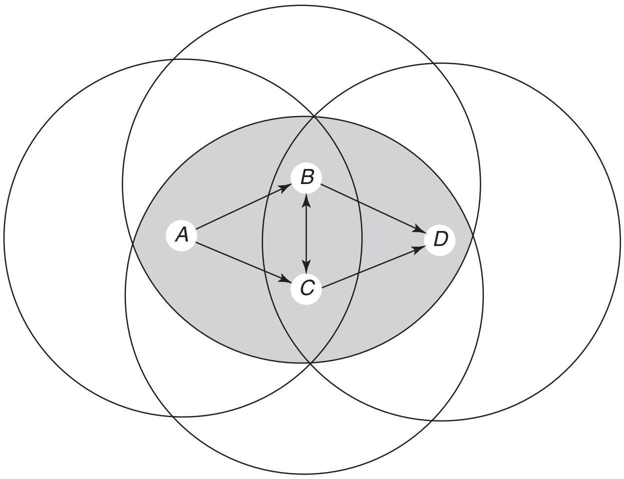


Figure 2.10 Collision with blind flooding.

Broadcasting protocols can also be further divided into three categories based on the information each node keeps: (i) no need of neighbor information; (ii) the one-hop neighbor information; (iii) the two-hop or more neighbor information. Neighbor information can be further divided into *position* and *topological* information, depending on whether or not nodes are aware of their absolute or relative geographic coordinates. Blind flooding does not require any neighbor information. Area-based beaconless broadcasting algorithm (Ovalle-Martinez *et al.*, 2006) also does not require neighbor information but needs the availability of geographic position of each node. Efficient flooding (Liu *et al.*, 2007) is based on the one-hop neighbor information. Multipoint relay protocol requires the two-hop topological neighbor information. The neighbor elimination scheme is based on either the one-hop information (if position information is available) or the two-hop topological information.

Some techniques for efficient broadcasting do not require any backbone beforehand. However, they can be considered as the process of backbone construction on the fly. They use the local information of nodes to eliminate redundant transmissions and thus decrease both energy consumption and the probability of collisions. Representative works are the neighbor elimination scheme, ABBA (Ovalle-Martinez *et al.*, 2006), Efficient Flooding scheme (Liu *et al.*, 2007) and tree-based broadcasting (Ding *et al.*, 2006). They will be described in following sections.

2.5.1 Neighbor Elimination Scheme

The *neighbor elimination scheme* was independently proposed by Peng and Lu (2000), Stojmenovic and Seddigh (2000), and Lim and Kim (2000) in August 2000. The basic idea is that a node receiving a message waits for a time-out duration before it retransmits the message. If all its neighbors are covered before the time-out expires, the node decides not to retransmit the message. Otherwise, if there are still uncovered neighbors, the node retransmits the message. The time-out could be randomly chosen or be computed based on several parameters. For example, it is desirable to let the nodes with larger degree retransmit the message sooner so that more nodes can be covered by one transmission. Thus, for example, the time-out could be defined as follows: $timeout = C/$(number of uncovered nodes), where C is a constant and the number of nodes that are not yet covered is based on one node's knowledge.

The neighbor elimination scheme is illustrated by Figure 2.11. Suppose each node has the one-hop knowledge (and the position information of neighbors). Node s initiates a broadcast and sends a message to its neighbors b, d, and g. Each of the three nodes determines its time-out duration assuming $C = 1$. Since each node has the one-hop knowledge, b knows that d has been covered by broadcasting of s (and vice versa). Therefore, nodes b, d, and g set the time-out duration to 1/2, 1/3 and 1/2, respectively, as shown in Figure 2.11a. Since node d has the smallest time-out duration, it transmits next and nodes a, c, and h receive the message. Node c sets its time-out to 1/2 since it has only two

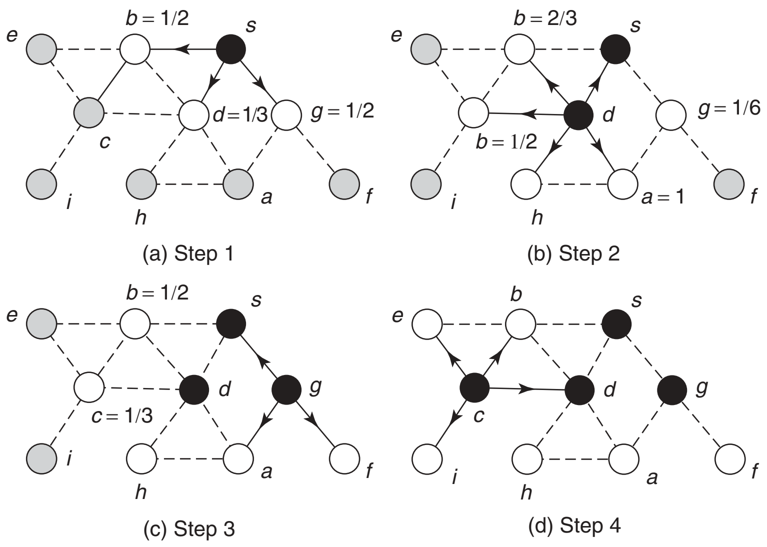


Figure 2.11 Neighbor elimination scheme. (a) Step 1, (b) Step 2, (c) Step 3, (d) Step 4.

neighbors which are not covered yet. Node a sets its time-out to $1 = 1/1$ since it is not aware that g has received the message. After transmitting from d, node g updates the time-out to be 1/6 since 1/3 time units have elapsed. Note that b knows c is covered by d, therefore it adjusts its initial time-out to be 1. Deducting the already elapsed time of 1/3, the updated time-out of d is 2/3, as shown in Figure 2.11b. At step 3 in Figure 2.11c, node g transmits the message to node f and the time-out duration of nodes a, b, and c are updated accordingly. At the last step in Figure 2.11d, node c forwards the message to node e and node i. Eventually, all nodes receive the message.

In the neighbor elimination-based algorithm, a node retransmits the message when the time-out expires if some of its neighbors (at least one) do not receive the message. However, these neighbors may receive the message from other nodes after the time-out. Thus, the retransmission of the node could be redundant. To reduce these redundant retransmissions, a *responsibility-based flooding scheme* (*RBS*) was proposed by Khabbazian and Bhargava (2008a). An RBS assumes that each node knows the location information of its one-hop neighbors. The basic idea of the scheme is that a node cancels its retransmission if it is not *responsible* for any of its neighbors. A node, say A, is not responsible for a neighbor, say B, if B has received the message or if there exists another neighbor, say C, such that C has received the message and B is closer to C than to A [$dist(B, C) < dist(B, A)$].

In the example in Figure 2.12, suppose node A receives a flooding message from node F. A learns that nodes E and D have received the message. Although nodes B and C have not received the message, node A will cancel its retransmission since $dist(B, E) < dist(B, A)$ and $dist(C, D) < dist(C, A)$.

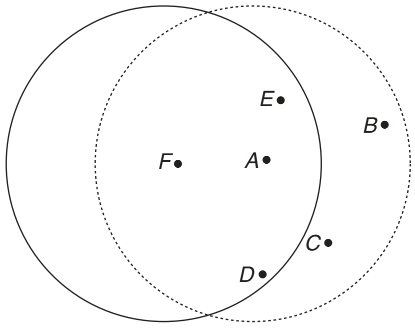


Figure 2.12 RBS: A does not retransmit message received from F.

2.5.2 Neighbor Elimination and Backbone-Based Broadcasting

The neighbor elimination scheme was further extended by Stojmenovic *et al.* (2002) by integrating the backbone technique. The framework is based on two concepts: CDSs as the particular type of backbone that provides reliability, and the neighbor elimination scheme. The connectivity of CDS provides propagation through the entire network while the domination of CDS guarantees that all of the nodes in the network will be reachable. The main goal is to provide reliable broadcasting with a minimal number of retransmissions. Suppose a CDS backbone has been constructed in a given network. The general algorithm for intelligent broadcasting is as follows (Stojmenovic *et al.*, 2002). Suppose a source node initiates a broadcast message. Upon receiving the message for the first time, a node will not rebroadcast it if the node is not in the CDS. If it is a member of the CDS, it will select a time-out period based on several parameters. During the time-out, the node may receive more copies of the message. It will update its forwarding list by eliminating all neighbors which have been covered by these message copies. When the time-out expires, the node will rebroadcast the message if its forwarding list is nonempty. Otherwise, the retransmission is canceled.

In the example in Figure 2.13, suppose node S is the source node and that $key = (degree, ID)$. Nodes B, D, E, F, S, G, H, J, and K have two unconnected neighbors. Since the neighbors of J are covered by H with a higher key value, J does not join the CDS. Note that the neighbors of B are covered by connected E and F which have larger key values. Thus, B does not join the CDS. Similarly, the neighbors of K are covered by F and H and neighbors of both D and S are covered by E and G. Ultimately, nodes E, F, G and H form a CDS. After receiving the message from S, node G sets its time-out to 1/3, since it has three uncovered neighbors (i.e., E, H, and J). When the time-out expires, G retransmits the message to its neighbors. Since node E is not aware of coverage of node D by S, E sets its time-out to 1/4 due to four uncovered neighbors A, B, D and F. H sets its time-out to 1/2 due to two uncovered neighbors F and K. Node E then retransmits the message because of the shorter time-out. After this

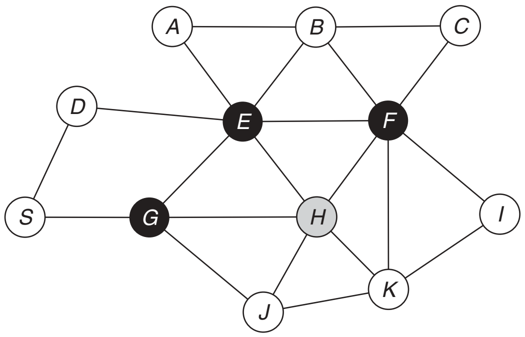


Figure 2.13 CDS nodes G, E, F retransmit while H does not in neighbor elimination and backbone-based broadcasting algorithm.

retransmission, H updates its time-out to 3/4 (adjust the original time-out to 1 and then deduct 1/4 time units that have already elapsed) and F sets its time-out to 1/3 due to three uncovered neighbors C, I and K. Since the time-out of F is shorter than that of H, F retransmits the message next. A CDS node H then cancels its retransmission and all nodes have received the message.

Note that CDS of a complete graph, by definition (Wu and Li, 1999; Stojmenovic *et al*., 2002), is empty. However, the DS is a set of nodes that retransmit the message, and in case of a complete graph, no retransmission (after the source sends the message) is needed. If a nonempty DS is indeed always preferred then the node with the largest key can be added to it. That is, nonintermediate nodes with larger key than any of their neighbors can be added to the DS. It will have no impact on broadcasting process because of neighbor elimination (Stojmenovic *et al*., 2002).

To increase the reliability at the medium access control (MAC) layer, Stojmenovic *et al* further introduced the retransmissions after negative acknowledgments (RANA) protocol (Stojmenovic *et al*., 2002). The collision of two messages normally occurs after the initial portion of the first message in which the sender's information is located has already been received. Therefore, the receiver can send a negative acknowledgment back to the sender to ask for retransmission.

2.5.3 Tree-Based Broadcasting in 802.15.4-Based Sensor Networks

Tree-based broadcasting in IEEE 802.15.4 and ZigBee networks was studied in Ding *et al*. (2006). It is assumed that the whole network is organized into a single tree, which then allows a convenient hierarchical addressing, based on ZigBee specification. The addresses of parent and children of a given node can be derived from its network address without any information exchange. Details of an address assignment in ZigBee networks can be found in http://www.zigbee.org.

Two tree-based broadcasting algorithms are proposed based on a hierarchical address space in ZigBee networks. In the *on-tree self-pruning rebroadcast* (OSR) algorithm, neighbor elimination is applied on-tree neighbors (parent and children) of each node, instead of all neighbors of the same node. Note that the two-hop neighbor information is not directly available in ZigBee networks, but can be derived from the hierarchical addresses of the one-hop parent and children nodes. In the *optimal on-tree forward node selection* (OOS) algorithm, the technique of MPR (explained in the next section) is applied. The forward node selection problem for ZigBee networks is reduced to finding the minimal forward node set to cover tree neighbors of all neighbors of a given node. Thus, partial two-hop neighbor knowledge is required, with the appropriate memory requirements. The *ZigBee on-tree selection* (ZOS) algorithm has a similar performance as the OOS, but consumes less memory space than OOS; the difference is that ZOS derives tree neighbors of a node only when necessary while OOS stores the information beforehand.

2.5.4 MPR-Based Broadcasting

Multipoint relay-based broadcasting was proposed in Qayyum *et al*. (2002) and three other articles independently (the remaining references can be found in Simplot-Ryl *et al*. (2005)). The set of relay points for a given node are piggybacked to the packet and transmitted as a forward list. An adjacent node that is requested to relay the packet again determines its forward list, and similarly transmits its neighbor's forward list along with the message. This process is iterated until the broadcast is completed. The methods differ in details on when and how a node determines its forward list. The general principle of forward list determination was outlined already in this section on *MPR*-based backbone. In the adaptation of multihop relaying presented in Peng and Lu (2000), the broadcasting node transmits list of its neighbors at time of broadcast packet transmission, not as part of any "hello" message. The two-hop neighbors knowledge is used to determine which neighbors also received the broadcast packet in the same transmission, and these nodes are already covered and are removed from the neighbor's graph used to choose the next hop relaying nodes. Finally, if a broadcast message is received from a node that is not listed as a neighbor, the message is retransmitted, to deal with high mobility issues. In CDS-based broadcast algorithm (Peng and Lu, 2000), the sender node establishes priorities between forwarding nodes, and each forwarding node should eliminate from consideration not only neighbors of the sender node, but also neighbors of each relaying node with higher priority.

Multipoint relay-based broadcasting, when compared to backbone-based broadcasting, has similar or a somewhat better performance in terms of rebroadcast savings, but has the message overhead due to inclusion of a forwarding list in the packet which may be significant for energy limited tiny sensors. A detailed study can be found in Ingelrest *et al*. (2007).

2.5.5 Area Coverage-Based Beaconless Broadcasting

Sensor area coverage technique was applied in Ovalle-Martinez *et al.* (2006) (the same algorithm was also described in the "Conclusion and future work" section of Shaikh *et al.* (2003); a similar idea appeared later in Heissenbüttel *et al.* (2006)) for the design of beaconless broadcasting protocol in *ad hoc* and sensor networks. It assumes that each node knows its geographic position. Distinguished from most existing broadcasting protocols, the proposed solutions are beaconless, that is, they do not require any neighbor knowledge for each node. Nodes decide whether or not to retransmit the broadcast message based only on the information obtained during the broadcast process. Ovalle-Martinez *et al.* (2006) proposed several ABBAs, for two-dimension and three-dimension spaces.

The proposed broadcasting protocol for two-dimension area (2D-ABBA) works as follows. Upon receiving the first copy of the broadcast message, a node sets a time-out before it may decide to retransmit the message. The node may receive several copies of the message from different senders before the time-out expires. Since the position information of senders is included in the broadcast messages, the node can determine whether or not its transmission area, modeled as a circle, is covered by these senders. If the transmission area of the node is fully covered before the time-out expires, the node stops the timer and decides not to retransmit the message. Otherwise, the node retransmits the broadcast message once the time-out expires.

Since all nodes have the same transmission radius, coverage of the transmission area is equivalent to the coverage of perimeter of the area. In the example in Figure 2.14a, suppose node A receives the broadcast message from node B, C, and D before its time-out expires. Since the perimeter of A is fully covered by the union of transmission area of B, C, and D, A decides not to retransmit the message. However, in Figure 2.14b, A finds its perimeter is not fully covered and decides to retransmit the message.

Two methods were proposed in Ovalle-Martinez *et al.* (2006) to set the time-out for each node. One is to set the time-out in reverse proportional to the length of uncovered portions of the perimeter. The other method simply uses a random function with values between 0 and 1.

The transmission area of a node is modeled as a sphere in a three-dimension area. In 3D-ABBA3 algorithm (Ovalle-Martinez *et al.*, 2006), the intersections of three spheres are considered for coverage criteria. Suppose the three spheres are centered at A, B, and C, respectively. The two intersection points of any three spheres are on their 3D perimeter. Suppose A receives the broadcast message from B and C. If each such intersection point is included inside another sphere centered at node D that also retransmitted the message, the transmission sphere of node A is fully covered. Details are given in Chapter 3 where this criterion has been used for sensor volume coverage.

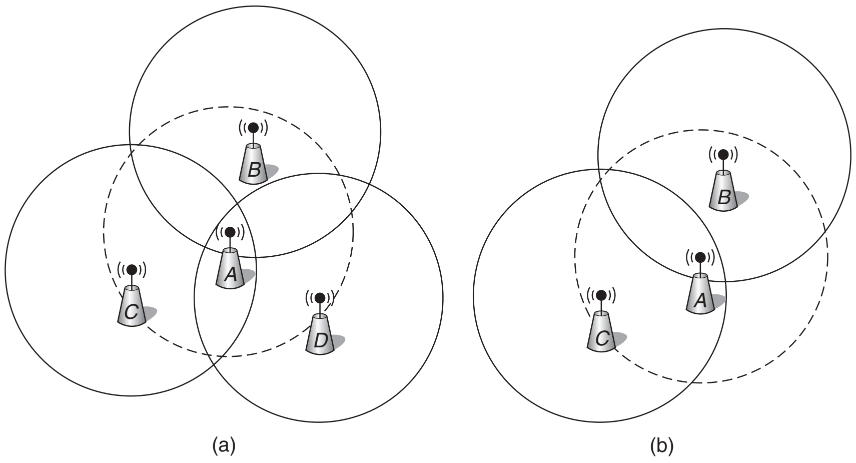


Figure 2.14 Coverage of perimeter of the transmission area.

2.5.6 Efficient Flooding and Slice-Based Broadcasting

Liu *et al.* (2007) presented a sufficient and necessary condition of guaranteed deliverability for any flooding scheme that is based on only the one-hop neighbor information. On the basis of the sufficient and necessary condition, a sender-based flooding scheme, called *efficient flooding*, was proposed. The neighbor's area of node s is the union of coverage disks of all s's neighbors plus s itself. Let $F(s)$ denote the set of forwarding nodes that is computed by node s.

Theorem 2.2. Liu *et al.* (2007) A one-hop flooding scheme achieves guaranteed deliverability if and only if for each node s the neighbor's area of s is covered by *F(s)*.

On the basis of Theorem 2.2, the efficient flooding scheme computes the minimal $F(s)$ for each forwarding node s. To minimize $F(s)$, every node in $F(s)$ must contribute to the neighbor's boundary of s. Otherwise the node can be removed from $F(s)$ without affecting the coverage area of $F(s)$.

Two algorithms are proposed in Liu *et al.* (2007) to compute the minimal $F(s)$. The first algorithm with time complexity $O(n^2)$ works as follows. All one-hop neighbors of node s are sorted in descending order into a list according to their Euclidean distance to s. The first node in the list, the farthest away from s, is included in $F(s)$ since it is sure to contribute to the boundary of neighbor's area of s. Each time, the next node in the list is added to $F(s)$ if its coverage disk is not fully covered by the so far constructed $F(s)$. The process continues until all nodes in the list are considered.

The second algorithm (Liu *et al.*, 2007) is to compute the neighbor's boundary of s, and thus the nodes that contribute to this boundary are the nodes in $F(s)$.

Its time complexity is $O(n \log n)$. The basic idea is to use the pair-wise boundary merging method to compute the boundary of the neighbor's area of s. Initially, each node is arbitrarily paired with another node to merge their coverage boundaries. Then, the merged pair's boundary is further merged with another pair's boundary. This merge operation is repeated until there is only one big merged boundary, which is the boundary of neighbor's area of s. The minimal $F(s)$ consists of the nodes that contribute to this boundary.

The source s first computes its $F(s)$ and then broadcasts a flooding message which attaches IDs of nodes in $F(s)$. Upon receiving the flooding message, each node u takes the same operation as the sender if its ID is included in the message and it is the first time for receiving the message. Otherwise, node u ignores the message. If the transmission is ideal, that is, no collision and no packet loss, all nodes in the network will ultimately receive the flooding message according to Theorem 2.2. Forwarding node optimization is further studied in Liu *et al.* (2007). A node in $F(s)$ does not need to retransmit the flooding message if all its neighbors have been covered by node s or other nodes in $F(s)$. Node IDs are used to break the tie if there is a loop in the optimization.

Khabbazian and Bhargava (2008b) proposed a slice-based broadcasting algorithm which is a sender-based scheme. Similar to (Liu *et al.*, 2007), it assumes that each node knows the location information of its one-hop neighbors. Each time, the node that holds the flooding message nominates a subset of its one-hop neighbors, that is, forwarding set, to forward the message. Upon receiving a flooding message, only nodes whose IDs are included in the message and have never retransmitted before will forward the message.

To select neighbors to forward the message, a node divides its communication region, modeled as a circle, into *bulged slices*. In the example in Figure 2.15a, there are three circles with the same radius and the center of each circle locates exactly at the intersection point of the other two circles. The slice with a bold border is defined as the bulged slice around node A. The forwarding set of node A is determined such that any nonempty bulged slice around A contains at least one node from the forwarding set. Suppose node A starts to compute its

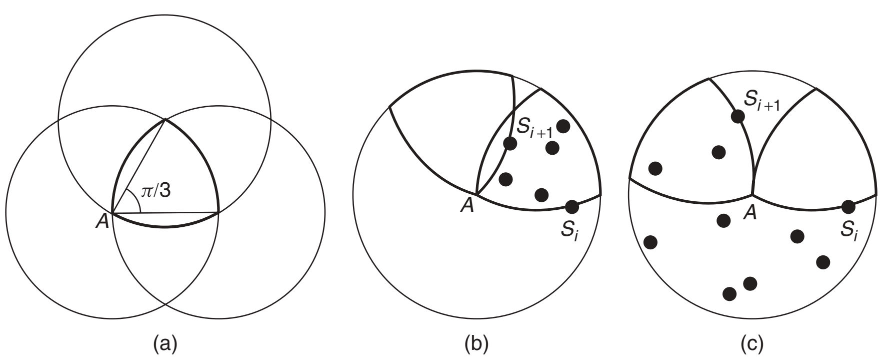


Figure 2.15 Slice-based broadcasting algorithm.

forward set. A randomly selects the first node S_1 from its neighbors. Suppose the current selected node is S_i. If the bulged slice of S_i is nonempty then the next node S_{i+1} is the farthest anticlockwise rotation of the bulged slice of S_i. In the example in Figure 2.15b, S_{i+1} is such node inside the bulged slice of S_i. If there is no node in the bulged slice of S_i then S_{i+1} is the nearest anticlockwise rotation, see Figure 2.15c for an example. The selection of forwarding set is complete at S_m if either S_m is inside the bulged slice of S_1 or S_1 is inside the bulged slice of S_m. Authors prove that the slice-based broadcasting algorithm can achieve 100% delivery in a collision-free network (Khabbazian and Bhargava, 2008b).

2.6 PHYSICAL LAYER-BASED FLOODING, NEIGHBOR DETECTION AND ROUTE DISCOVERY

2.6.1 Route Discovery with Realistic Physical Layer

In reactive routing protocols, such as dynamic source routing (DSR) (Johnson *et al.*, 1996) and *ad hoc* on-demand distance vector (AODV) (Perkins *et al.*, 1999), each node receiving a route discovery message will retransmit the message if a better route, for example. in terms of hop count and power consumption, is found. However, each node retransmits the route discovery message *at most once*. An important neighbor may miss the message under the given realistic physical layer, and thus either the route cannot be found or the constructed route may be far from the optimal one. Therefore, a single transmission cannot guarantee to reach each potential neighbor which may provide the best or the only route. Stojmenovic *et al.* (2005a,b) proposed design guidelines for routing protocols with a realistic physical layer. They suggested that each node could retransmit the given route discovery message *several times* rather than once as in the UDG model. Such multiple retransmissions may also serve to measure or reevaluate the packet reception probability. The basic idea is as follows. Upon receiving a route discovery message, a node checks if the received message contains information about a better route. If so, the node stops retransmission of the previously known best route, if it is still ongoing, and retransmits the new route discovery with a fresh counter. The optimal number of retransmissions depends on the network density. In very dense networks, a single retransmission of route discovery may be sufficient to find the best route. In sparse networks, a large number of retransmissions might be needed to discover important bridges/neighbors in the network.

The metric used in the route discovery process is based on assumptions. For example, assuming communication is done with a hop by hop acknowledgement, the appropriate metric for route discovery is the expected hop count (EHC). ETX is proposed by De Couto *et al.* (2003) and counts all (re)transmissions, possibly acknowledgements between two nodes on a link. Thus EHC includes the expected number of retransmissions and the expected number of acknowledgments.

2.6.2 Neighbor Detection and Flooding with Realistic Physical Layer

In the UDG model, information between neighbors is exchanged by "hello" messages. After a node broadcasts a "hello" message, all of its neighbors receive it. However, if a realistic physical layer is adopted, each neighbor receives the message with certain probability that depends on distance and other factors. Some "hello" messages are not received and thus nodes cannot properly update information on their neighbors.

Goel *et al.* (2008) studied gathering neighborhood information with a realistic physical layer. Two neighbor detection protocols were proposed. In the first protocol, referred to as the *s-hello* protocol, each node sends exactly *s*-"hello" messages to its neighbors. With respect to the performance of localized position-based routing protocols, simulation results show that the gains in neighbor knowledge become limited for $s > 5$. In the second protocol, referred to as the *target density* protocol, the objective of each node is to learn about at least *td* (target density) neighbors, such that the routing protocol achieves good performance. Each node keeps sending "hello" messages until it receives *td* messages from its neighbors or until a time-out expires. In the case of uniformly distributed networks, nodes at the boundary of the network have to send a larger amount of "hello" messages than the nodes inside the network in order to achieve the target density. The main advantage of the target density protocol is that it saves a unnecessary "hello" message after the desired number of neighbors are found. The *s*-hello protocol on the other hand, blindly keeps sending messages possibly with too many or too little retransmissions.

The DS concept can be redefined based on the probability for receiving a packet. This leads to a definition and construction of DSs with the realistic physical layer. Brief details can be found in Simplot-Ryl *et al.* (2005), and the concept still has not been elaborated sufficiently.

2.7 PARAMETERLESS BROADCASTING FOR DELAY TOLERANT-NETWORKS

Existing broadcasting protocols can be CDS-based protocols for static networks, blind flooding for moderately mobile networks and Hyper-flooding (Viswanath and Obraczka, 2002) for highly mobile and frequently partitioned networks. In static networks, nodes do not move or they move slowly, such that network topology does not change during broadcasting. However, the line separating moderately mobile networks and highly mobile networks is often thin. Generally speaking, the difference is based on the percentage of neighbor changes for a node during broadcasting. A network is moderately mobile if each node is either static or moderately mobile. Thus, a network is moderately mobile even if there is a single moderately mobile node among static nodes in the network. A network is highly mobile if at least one of its nodes is highly mobile. Therefore, a network with a few high-speed vehicles and a lot of pedestrians along the road is highly mobile (Khan *et al.*, 2008).

Broadcasting protocols for static networks have been discussed in previous sections. Since the maintenance of neighbor knowledge is expensive or even impossible in mobile networks, blind flooding can still be applied in moderately mobile networks. However, it may not suffice in networks with temporary partitions and high mobility. This is due to the fact that each node in the network retransmits the broadcasting message only once and stops retransmission even if it discovers a new neighbor afterwards. *Hyper-flooding*, proposed by Viswanath and Obraczka (2002), addresses such scenarios. In hyper-flooding, a node will retransmit the broadcasting message whenever it discovers a new neighbor. Thus, it is able to disseminate messages across partitions when nodes move during broadcasting. It increases reliability at the cost of a high message overhead. An adaptive approach was proposed in Viswanath and Obraczka (2002) for both static and mobile networks, based on using some thresholds. It has been pointed out in Khan *et al.* (2008) that the threshold is based on the parameters for mobility and traffic which may be difficult or impossible to gather.

Khan *et al.* (2008) studied broadcasting without parameters in static to highly mobile networks. It is suitable for applications in delay tolerant-networks. The proposed protocol, PBSM (parameterless broadcasting from static to mobile), assumes that each node has the two-hop topological knowledge. The goal of the protocol is to adapt itself to any static/mobility scenario automatically without calculating any threshold kind of parameter. The protocol combines CDS technique and the neighbor elimination concept. At the initialization stage, nodes periodically exchange "hello" messages in order to gain the two-hop knowledge. A CDS is calculated after each "hello" message round. Each node in the protocol maintains two lists. One of them, denoted by R, records the neighbors that have received the messages. The other, denoted by N, records the neighbors that did not receive the message yet. Each node sets a time-out period. The nodes in CDS select shorter time-outs than the nodes outside CDS, such that the nodes in CDS have a higher chance to retransmit the broadcasting message. During the waiting period, lists R and N are updated upon the receipt of each message copy. When the waiting period expires, the node retransmits if N is nonempty. The message is memorized until T "hello" messages are received. N and R are updated for each "hello" message received. Nodes that are no longer one-hop neighbors are removed from these lists while new neighbors are added if discovered.

Using example in Figure 2.13, node J will also retransmit since it is not aware of covering of K by F. Improvement is possible by adding a responsibility-like concept. J is aware that H from CDS canceled transmission, which was possible only if H is aware that some other node covered K. Such modification can be added to the algorithm (Khan *et al.*, 2008).

Example in Figure 2.16 shows how the protocol works in a mobile network where the network is initially partitioned into two components (position 1A of node 1). Flooding started from node 0, and all nodes in the left partition receive the message, and flooding or CDS-based broadcasting stops. When node 1 moves

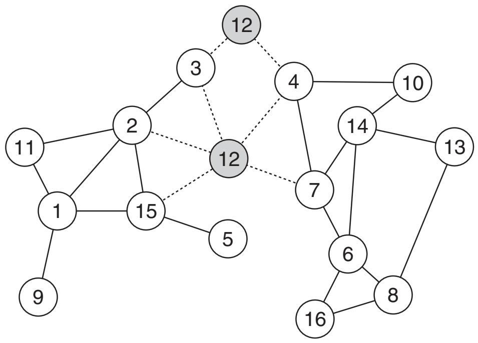


Figure 2.16 A mobile network.

to positions 1B or 1C, flooding is not reactivated in these two methods, and the right portion of the network does not receive the message. In the hyper-flooding algorithm (Viswanath and Obraczka, 2002), nodes from the left partition systematically inform node 1B (and later 1C) as they meet it for the first time. Also, node 1 retransmits the message any time it sees a new neighborhood on the right partition, where receiving nodes similarly activate flooding in that partition. This, overall, amounts to a significant inefficiency of retransmitting messages. Parameterless broadcasting from static to mobile (Khan *et al.*, 2008) will let node 3 retransmit after discovering node 1 (at position 1B) since node 3 then becomes a CDS member (bridging two partitions) and its list N becomes nonempty. Retransmission is similarly done by node 1 and the nodes in the right partition receive the message. During later movement from position 1B to position 1C, only node 1 is new in the neighborhood, so nodes on both sides do not retransmit the message possibly sent by moving node 1 going into a new neighborhood.

2.8 BACKBONES AND BROADCASTING IN SENSOR–ACTUATOR NETWORKS

Different from sensor networks or *ad hoc* networks, sensor actuator networks are normally heterogeneous networks where actuators are more powerful than sensors, in terms of the energy level, communication and computing capabilities and memory. Therefore, actuators are expected to take more responsibility and play more important roles in backbone construction and broadcasting. One existing solution described below uses access points instead of actuators, while the other is based on higher key assigned to actuators.

2.8.1 Broadcasting for Hybrid *ad hoc* and Sensor Networks

Routing and broadcasting for hybrid *ad hoc* and sensor networks was studied by Ingelrest *et al.* (2006a). It assumes that there are two types of nodes in the hybrid network: mobile nodes and fixed access points. The access points are more powerful than mobile nodes in terms of computational ability, memory, and energy supply. The mobiles can either directly connect to an access point, or use *ad hoc* mode to reach the access point via multiple hops. Figure 2.17 shows an example of such a hybrid network. There could be wired or wireless connection between access points P_1 and P_2. Mobile nodes a, b, c, f are directly connected to the access points while mobile nodes e and d require a relay of other mobile nodes to connect the access points.

Component neighbor elimination-based flooding was proposed in Ingelrest *et al.* (2006a). It divides the network into components, one of which consists of an access point and the mobile nodes attached to the access point (one or multiple hops). Since the energy consumption of access points is not considered, the flooding protocol is based on the observation that transmission via access points is more efficient than that via mobile nodes. The basic idea of the protocol is as follows. An access point retransmits a flooding message immediately once it receives the message for the first time. A mobile node that receives the message for the first time sets a time-out and monitors its neighborhood. It cancels the relay if all its neighbors have received the message when the time-out expires. Otherwise, it retransmits the message. The protocol assumes a sparse connected structure of access points, if they use wireless links between them. Otherwise, it can be modified by applying CDS-based broadcasting discussed in this chapter. If access points use wired links among themselves then appropriate link-based flooding among them needs to be used (this is out of the scope of this book).

The drawback of component neighbor elimination-based flooding is its increased latency since mobile nodes retransmits the message after its time-out expires. It was pointed out in Ingelrest *et al.* (2006a) that there are basically two methods to transmit the message. One is "*ad hoc* mode" in which the message is transmitted via mobile nodes without passing through any access point. The other is "access point mode" in which the message is transmitted from one component to the destination component via access points. *Adaptive flooding* (Ingelrest *et al.*, 2006a) was proposed to minimize the latency of the broadcast by adaptively selecting the shorter path between these two methods. In the example in Figure 2.17, suppose node e receives a message initiated by node b. If the message has not passed any access point, node e decides to relay the message if there exists a neighbor f and $hc(e, f) + 1 < hc(b, P_1) + hc(f, P_2)$, where $hc(a, b)$ denotes the hop count between nodes a and b. If the message has passed an access point, each node relays the message to its neighbors within the same component.

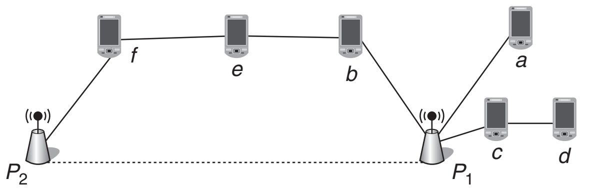


Figure 2.17 A hybrid network with two access points.

The MPR technique can be easily generalized to hybrid networks (Ingelrest *et al.*, 2006a). When considering which neighbors should relay the message, access points should be first added to the MPR set and mobile nodes are included only when necessary.

2.8.2 Dominating Set-Based Backbone

Suppose V is the set of sensor nodes in the network. *k-Dominating Set* (*k-DS*) is a subset of V such that each node of V is either a member in k-DS or can reach at least one node in k-DS within k hops. *k-Independent dominating set* (*k-IDS*) is a k-DS such that any two dominators are within at least k-hop distance of each other in the set. McLaughlan and Akkaya (2007) proposed an algorithm to position actuators as CHs such that the number of covered sensors is maximized and data gathering time is minimized. It is done by finding a k-hop IDS (k-IDS) of the underlying sensor network, and placing actuators at near selected sensors for k-IDS. Sensor nodes join its dominator based on the distance to and ID of the dominator. A weight function is used to prefer certain nodes that have many k-hop neighbors and are farther from borders of other clusters. Finally, each actuator acts as a CH in its cluster and all sensor nodes in each cluster can reach their CH within k hops.

A CDS-based backbone is not necessarily based on UDG. It can be applied in hybrid networks such as sensor and actuator networks. For instance, the generalized coverage-based algorithm (Stojmenovic *et al.*, 2002, 2004; Dai and Wu, 2003; Carle and Simplot-Ryl, 2004), introduced in Section 2.4.3, can be extended to hybrid networks. Note that in the self-pruning rule of the algorithm (Dai and Wu, 2003), a key of nodes is used to determine the priority. To adapt it to hybrid networks, the key of a node can be extended to include the energy level of the node: $key(s) = \{E_s, \text{ID}(s)\}$, where E_s is the energy level of the node similarly as in algorithm in Wu *et al.* (2002). Since access points have a higher energy level than mobile nodes, they will always be selected as dominators and thus will be part of the broadcasting process. Such a model could be applied to sensor and actuator networks where sensors and actuators are treated as mobile nodes (although they are normally static) and access points, respectively.

In the example in Figure 2.18, suppose node 0 is an access point and other nodes are mobile nodes. After applying the algorithm in Dai and Wu (2003), the

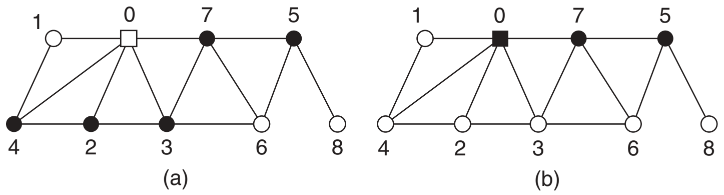


Figure 2.18 CDS construction with different key selections.

resulting CDS is shown in Figure 2.18a. If the key $\{E_s, \mathrm{ID}(s)\}$ is employed, the access point 0 is selected into the DS and the result is shown in Figure 2.18b.

2.9 RNG AND LMST

The next section on minimal energy broadcasting is based on some geometric structures. These structures will be described in this section. Basic familiarity with minimal spanning tree (MST) is assumed. Minimal spanning tree for a weighted graph with n nodes is a minimal connected graph containing these n nodes as vertices. Minimal spanning tree is a tree since otherwise the longest edge in any cycle can be removed without affecting connectivity. This observation is used in Kruskal's algorithm (Kruskal, 1956) to construct MST. All edges are sorted in increasing order, and considered for inclusion in MST in that order. The candidate edge is included in MST if it does not create a cycle in an already constructed portion of the MST.

Relative neighborhood graph (RNG) and localized minimal spanning tree (LMST) are two well-known *planar* graphs that are utilized in various topology constructions. A planar graph is a graph, which can be drawn on the plane in such a way that the edges intersect only at their end points. Figure 2.19 gives examples of an RNG and an LMST and illustrates also planar graphs. All nondashed edges form RNG while bold edges form LMST. The dashed edges are remaining edges in UDG. The edge (1, 4) belongs to RNG while it does not belong to LMST since the edge is not in MST of the one-hop subgraph of nodes 1 (will be explained next).

An edge uv is in RNG (first defined in Toussaint (1980)) if it is not the longest edge in any triangle uvw where w is any third node from the set. In

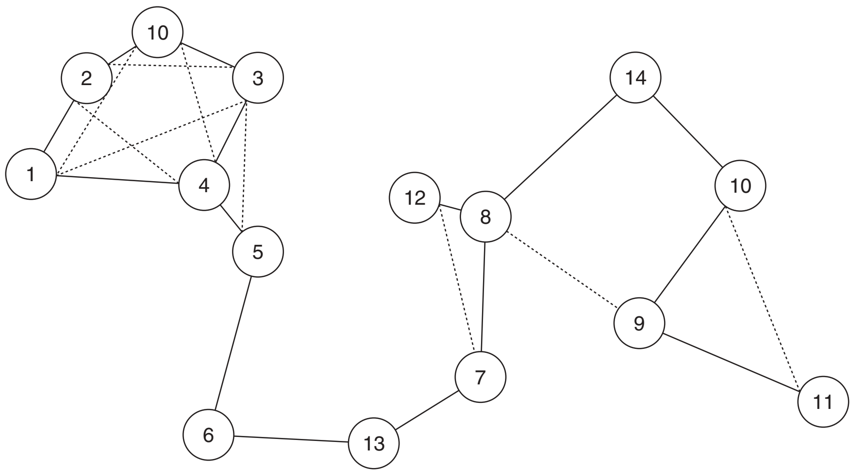


Figure 2.19 LMST (bold edges) and RNG (all nondashed edges) of an unit disk graph (all edges).

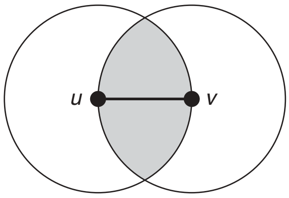


Figure 2.20 Forbidden region of edge uv in RNG

Figure 2.20, the edge uv belongs to RNG if there are no other nodes in the gray area, also referred to as the *forbidden region*. For any point w in the forbidden region, edge uv is the longest edge in triangle uvw. Once a node gathers location information of all its neighbors (via beacon messages), it can determine for each of its edges, if there is another neighbor which is included in the forbidden region, using the above criterion. Construction of RNG therefore does not require any additional messages. The average degree of RNG is about 2.5 (Hou *et al.*, 2005). Planarity of RNG follows from planarity of its superset Gabriel graph, which will be described and proven in Chapter 4.

In LMST (Li *et al.*, 2003), each node u is assumed to collect position information of its one-hop neighbors. Node u then computes the MST of its one-hop neighbors subgraph $N(u)$. An edge uv belongs to LMST if and only if uv is in both MST($N(u)$) and MST($N(v)$). To make this decision, a message exchange between neighbors (in addition to beacon message to learn neighbors) is required. The average node degree of LMST is about 2.04 (Hou *et al.*, 2005). Li *et al.* (2004) showed that LMST is a planar graph (it also follows from planarity of its superset RNG). They extended LMST to k-hop neighbors. That is, each node knows positions of its k-hop neighbors and LMST is constructed based on the k-hop subgraph of each node.

Theorem 2.3. (Ovalle-Martinez *et al.*, 2004; Cartigny *et al.*, 2005) MST $\subseteq$ LMST $\subseteq$ RNG.

Proof. We first give proof of MST $\subseteq$ LMST (Ovalle-Martinez *et al.*, 2004), then the proof of LMST $\subseteq$ RNG (Cartigny *et al.*, 2005). Both theorems are proven by contradiction.

Suppose there exist edges that belong to MST but do not belong to LMST. Let e be the shortest such edge. Suppose Kruskal's algorithm (Kruskal, 1956) is used to construct MST in subgraph *N(u)* of each node u. That is, edges are considered one by one in the increasing order of their length. When e is considered, since it is not included in LMST, it creates a cycle C in LMST with e being the longest edge in the cycle (Figure 2.21). Some edges from C are not in MST (otherwise there is a cycle since e belongs to MST). Consider now expanded cycle C' constructed from C as follow. Let f be an edge from C that is not in MST. Adding f into MST creates a cycle B with f being the longest edge in the cycle. The cycle consists of f and a path consisting of edges from MST. Replace f in C with all the edges from that path. Each such replacement enlarges the cycle

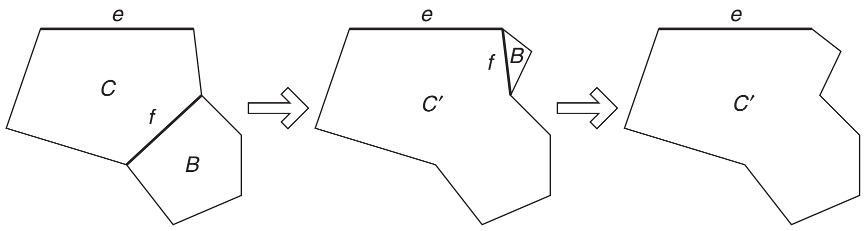


Figure 2.21 Proof by contradiction that MST $\subseteq$ LMST: e from MST becomes the longest edge in a cycle.

C, but does not add any edge longer than f, and consequently longer than e. This replacement process can continue, enlarging C' in each step. Finally, after replacing all non-MST edges with the corresponding paths of MST edges, edge e remains the longest edge of C'. But all the other edges of C' are now also in MST. It is a contradiction since MST has no cycles. Therefore, MST $\subseteq$ LMST.

Suppose that there exists an edge uv such that $uv \in$ LMST and $uv \notin$ RNG. Since $uv \notin$ RNG, there must exist a node $w \in N(u) \cap N(v)$ and uv is the longest edge in triangle uvw. Since $uv \in$ LMST, either uw or vw is not in LMST (otherwise, there is a cycle and it is not a tree). Without loss of generality, suppose uw is not in LMST. Edge uv could be replaced with uw in MST($N(u)$) and the resulting overall weight in MST is lower than the original one, which is a contradiction. Therefore, LMST $\subseteq$ RNG. ♦

Note that the degree of each node u in RNG (and therefore in LMST) can be bounded (normally to 5, exceptionally to 6) but only if lengths of each edge are distinct (otherwise there exist examples where an edge in RNG has very large degree). This can be achieved easily by considering record ($|uv|$, min (key(u), key(v)), max(key(u), key(v)), as the new “length” of an edge uv in the above definition.

2.10 MINIMAL ENERGY BROADCASTING

Broadcasting was discussed so far with the hop count as the metric of their performance. An alternative metric, often used in the literature, is the power for transmission and reception between nodes. It can be used if nodes can adjust their transmission powers, which we assume for this section. The problem of selecting forwarding nodes, and their transmission radii, so that all nodes in the network receive the message, and the total power used for these transmissions is minimized, is known as the *minimal energy broadcasting* problem. Detailed surveys of existing solutions can be found in Ingelrest *et al*. (2005) and Liu *et al*. (2005) (see also Stojmenovic *et al*., 2007 for few additional solutions).

Wieselthier *et al*. (2000) described centralized broadcast incremental power (BIP) algorithm. It is the most popular among several dozens of existing centralized solutions. In BIP, nodes are added one at a time into an existing growing tree, so that the additional power at each (incremental) step is minimized. There are two options: to add new transmission, or to increase transmission radius of existing transmitting node, so that a new node is added with the smallest possible additional energy. broadcast incremental power and other proposed methods in practice behave like MST since $2 \leq \alpha$ in power metric model $p(d) = d^{\alpha} + c$, for link at distance d, unless c is significant with respect to network density. The major disadvantage of centralized algorithms is their communication overhead to gather and maintain a global network view at each node. This overhead is not a part of BIP (Wieselthier *et al.*, 2000) and all other proposed centralized solutions.

Ingelrest and Simplot-Ryl (2008) described a localized version of BIP algorithm, called localized broadcast incremental power (LBIP). Each node maintains a two-hop information instead of network wide information. Sender node makes BIP decisions for all nodes in its two-hop neighborhood. It attaches a list of records (A, R_A), where A is the one-hop neighbor, and $R_A > 0$ is a nonzero transmission radius to be applied by neighbor A, with the message it transmits for broadcasting. No instruction is given to neighboring nodes designated as passive. Nodes receiving a broadcast message for the first time will first check if they are designated to retransmit. If so, they apply their own two-hop information to in turn calculate the transmission radii for their neighbors. It may increase its transmission radius assigned by incoming message, possibly increase the transmission radii of some of its designated neighbors, but not decrease it. That is, they start from the partial BIP construction already made, and add their two-hop neighbors, not "seen" by a sender node, to that tree. Note that several neighbors may do such a calculation in parallel. The first of them to transmit will then make an impact on the other one, to adjust its result before its decision is announced. If a node receives several instructions with various radii for same node, itself or neighbor, the larger one will be always taken. Surprisingly, the performance of this algorithm, in terms of total energy used, is very close to the performance of centralized BIP protocol, while preserving localized behavior. The algorithm resembles the behavior of MPR-based broadcasting algorithm, which also includes a set of retransmitting neighbors with the message. This appears to be the main disadvantage of the LBIP algorithm, since the increased message size will impact the algorithm at medium access level, where longer messages increase the chance of collisions. The problem is especially notable when broadcasting relatively short messages, since the added length to the message has a larger percentage of the overall size. This is a very similar problem noticed when MPR broadcasting was compared to the CDS-based one in Ingelrest *et al.* (2007). Another drawback of LBIP, when compared to some other protocols mentioned here, is that it requires gathering the two-hop neighbor information instead of the one-hop.

We will now describe existing localized algorithms for minimum energy broadcasting problem, which do not increase the size of messages. For small values of constant c, (with respect to other constant α) MST is an optimal tree, but requires global information. Given a sparse graph that connects all nodes, localized algorithms [the first one was given by Cartigny, Simplot and Stojmenovic in

Cartigny *et al.* (2003)] are based on each node deciding its transmission radius so that all its neighbors in the selected structure are covered by existing transmissions. Some nodes may not transmit at all, when neighbor elimination-based techniques is applied (if a node discovers that all its neighbors in a selected structure are already covered by existing transmissions, it does not need to transmit at all).

Consider an example in Figure 2.22, where MST is used as a sparse structure. A message from source S is sent with a radius equal to the furthest neighbor C in that structure. It is also received by A and B. Node A needs to cover the remaining neighbor D and thus chooses transmission radius $|AD|$. Node B similarly covers the node C by radius $|BE|$. Node D transmits with radius $|DK|$, which in turn covers C by radius $|KC|$ (node K is not aware of transmission from node S already covering C). Node E covers its neighbor F, while its neighbor F chooses a radius $|FH|$ which is longest toward the remaining neighbors G, J and F in the structure. Node G does not need to retransmit since all its neighbors are covered by existing transmissions, like the one from F (neighbor elimination applied). Finally, node H covers neighbor I. Leaf nodes do not retransmit.

To make an algorithm localized, MST is replaced by RNG in Cartigny *et al.* (2003) (algorithm RBOP), and by LMST Cartigny *et al.* (2005) (algorithm localized broadcast oriented protocol (LBOP)). RNG structure, when replacing MST, leads to about twice more energy, while an LMST structure has about 50% additional energy. Relatively few additional edges in LMST prove to be costly with respect to energy since they are long edges.

For large c with respect to α (in power metric $r^\alpha + c$), many transmissions over short edges become energy expensive due to multiples of c. A single larger circle centered at a node, covering many one-hop neighbors which are not neighbors in a selected sparse structure, may be preferred. Large radius is also power expensive. Therefore, one can expect a trade-off in the size of covering circles. In Ingelrest *et al.* (2006b), it is shown that if a node decides to retransmit, the optimal transmission radius should be about $(2\,c/(\alpha - 2))^{1/\alpha}$, increased if necessary to preserve the desired connectivity. The formula was derived by

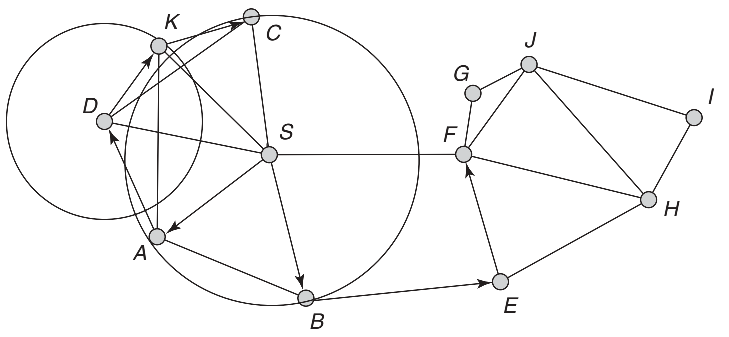


Figure 2.22 Covering neighbors in sparse structure for minimal energy broadcasting.

considering energy consumption in a regular honeycomb mesh with a variable edge length over a fixed area, and finding the edge length where minimum is achieved. Another proof, based on cost to progress paradigm, was given in Stojmenovic (2006). The cost of transmitting a packet is proportional to $r^{\alpha} + c$, while the progress made is proportional to the area being covered (i.e., r^2). The ratio $(r^{\alpha} + c)/r^2$ is minimized for $r = (2c/(\alpha - 2))^{1/\alpha}$.

A target radius localized broadcast oriented protocol (TR-LBOP) by Ingelrest *et al.* (2006b) works as follows. Each node manages two lists of neighbors, *L* and *L'*. *L* contains LMST neighbors. *L'* contains the rest of neighborhood. At the end of the neighbor elimination time-out period (during which node has listened to transmissions from some of its neighbors), if *L* is empty, the retransmission is canceled. Otherwise, the radius is chosen long enough to reach the furthest neighbor in *L*, and the neighbor in *L'* that is the nearest one to the target radius as set in the mentioned formula. The energy overhead of TR-LBOP with respect to BIP remains below 50% for all densities.

Target radius localized broadcast oriented protocol is further improved in Ingelrest *et al.* (2006b) to allow some nodes to be placed to sleep modes, while performing similarly to TR-LBOP in energy requirements. First, a subgraph where each node considers only neighbors whose distance is no greater than the *target radius* and neighbors in *RNG* or *LMST* is constructed. A CDS is then constructed using this subgraph. Next, nodes not selected for CDS are sent to the sleep mode (they periodically wake up for sending and receiving messages from associated closest DS nodes). Nodes in selected CDS remain active and apply TR-LBOP. Each node chooses a radius that covers its LMST neighbors and the target radius. The generalized covering rule for CDS is applied. Each passive node is attached to its nearest dominant neighbor. This may be referred to as algorithm TR-LBOP-D.

Chen *et al.* (2003) proposed PABLO algorithm, which needs two-hop topological information. Each node A verifies whether, for each of its neighbors B, there exists a common neighbor C so that $p(AC) + p(CB) < p(AC)$. All such neighbors B are eliminated. Among remaining neighbors, node A selects the furthest one, and uses it as the transmission radius. More precisely, it selects the largest power needed to reach any of the remaining neighbors. The authors (Chen *et al.*, 2003) cite RBOP (Cartigny *et al.*, 2003) which, instead of comparison $p(AC) + p(CB) < p(AC)$, uses comparison $\max(|AC|,|BC|) < |AB|)$ (from RNG definition), where $|XY|$ is the length of edge XY. Both PABLO (Chen *et al.*, 2003) and RBOP (Cartigny *et al.*, 2003) does not work well for large values of c and dense networks.

Wang *et al.* (2004) proposed the local shortest path tree (LSPT)-based topology for power aware routing and broadcasting in *ad hoc* and sensor networks. The weight of an edge between two nodes at distance d is $p(d) = d^{\alpha} + c$. Each node u applies Dijkstra's shortest path algorithm to find the shortest weighted path to each of its neighboring node, using only its local one-hop information (the concept can be extended to k-hop knowledge). Node u then keeps only edges that are going out of it in this structure, and removes the others. The result is sent to its neighbors. Each node then removes unidirectional links and adjusts its transmission radius according to the remaining logical links. Stojmenovic *et al.* (2007) proposed to use LSPT as the connected structure in minimal energy broadcasting algorithms, instead of LMST or RNG. This approach will automatically consider the impact of variable $c > 0$ since longer directed links can be more efficient than a multihop path toward them and this happens approximately up to an optimal target radius. Alternatively, one can simply apply a target radius (one option is $(2c/(\alpha - 2))^{1/\alpha}$) in the protocol that would correspond to TR-LBOP from (Ingelrest *et al.*, 2006b).

A number of articles assume $c = 0$ in the energy consumption model. For example, (Chiganmi *et al.*, 2008) described two such algorithms. In the Inside-OUT Power (INOP) adaptive approach, retransmitting a node's shortest weighted paths toward each neighbor, and selects transmission power equal to the distance to the farthest neighbor to whom the direct transmission is more power efficient than indirect transmissions via any neighbor. Only neighbors that are believed not to be covered by previous transmissions are considered. Each node makes its own decision in INOP, after a backoff timer. In other words, INOP is similar to RBOP (Cartigny *et al.*, 2003) and LBOP (Cartigny *et al.*, 2005), with LSPT (Wang *et al.*, 2004) replacing RNG and LMST as the connected structure (also independently proposed in Stojmenovic *et al.* (2007) for arbitrary value of c). PABLO (Chen *et al.*, 2003) can be considered as a version where a path with two hops only are considered instead of an arbitrary path lengths between two neighbors. The INOP-1 algorithm (Chiganmi *et al.*, 2008) is based on a neighbor designation similar to BIP over two-hop neighbors; that is, it is similar to LBIP (Ingelrest and Simplot-Ryl, 2008) restricted to the case $c = 0$.

An adaptive localized scheme for energy-efficient broadcasting in *ad hoc* networks with directional antennas was presented in Cartigny *et al.* (2004). It follows presented ideas on target radius-based retransmission, where the desired radius also depends on the antenna angles.

REFERENCES

Adjih C, Jacquet P, Viennot L. "Computing connected dominating sets with multipoint relays". Ad Hoc & Sens Wireless Netw 2005;1(1–2):27–39.

Alzoubi KM, Wan PJ, Frieder O. "Message-optimal connected dominating sets in mobile Ad hoc networks". Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002); 2002. pp. 157–164.

Basagni S. "Distributed clustering for ad hoc networks". Proceedings of the 1999 International Symposium on Parallel Architectures Algorithms, and Networks (ISPAN 99); 1999. pp. 310–315.

Basagni S, Carosi A, Petrioli C. "Sensor-DMAC: dynamic topology control for wireless sensor networks". Proceedings of IEEE Vehicular Technology Conference 2004-Fall (VTC 2004); Volume 4; 2004. pp. 2930–2935.

Basagni S, Mastrogiovanni M, Panconesi A, Petrioli C. "Localized protocols for ad hoc clustering and backbone formation: a performance study". IEEE Trans Parallel Distrib Syst 2006;17(4):292–306.

Carle J, Simplot-Ryl D. “Energy efficient area monitoring for sensor networks”. IEEE Comput 2004;37(2):40–46.

Cartigny J, Ingelrest F, Simplot-Ryl D, Stojmenovic I. “Localized LMST and RNG based minimum energy broadcast protocols in ad hoc networks”. Ad Hoc Netw 2005;3(1):1–16.

Cartigny J, Simplot D, Stojmenovic I. “Localized minimum-energy broadcasting in ad hoc networks”. Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003); 2003. pp. 2210–2217.

Cartigny J, Simplot D, Stojmenovic I. “An adaptive localized scheme for energy-efficient broadcasting in ad hoc networks with directional antennas”. Proceedings of 9th IFIP International Conference on Personal Wireless Communications PWC, LNCS 3260; 2004. pp. 399–413.

Chen X, Faloutsos M, Krishnamurthy SV. “Power adaptive broadcasting with local information in Ad hoc networks”. Proceedings of 11th IEEE International Conference on Network Protocols (ICNP); 2003. pp. 168–178.

Cheng X, Ding M, Du DH, Jia X. “On the construction of connected dominating set in Ad hoc wireless networks”. Special issue on Ad Hoc Netw Wireless Commun Mobile Comput 2004.

Cheng X, Huang X, Li D, Wu W, Du DZ. “Polynomial-time approximation scheme for minimum connected dominating set in Ad hoc networks”. Networks 2003;42(4):202–208.

Chiganmi A, Baysan M, Sarac K, Prakash R. “Variable power broadcast using local information in ad hoc networks”. Ad Hoc Netw 2008;6:675–695.

Cidon I, Mokryn O. “Propagation and leader election in multihop broadcast environment”. Proceedings of International Symposium on Distributed Computing, LNCS 1499; 1998. pp. 104–119.

Clark BN, Colbourn CJ, Johnson DS. “Unit disk graphs”. Discrete Math 1990;86:165–177.

Dai F, Wu J. “Distributed dominant pruning in Ad hoc networks”. Proceedings of the IEEE International Conference on Communications (ICC’03); Volume 1; 2003. pp. 353–357.

Dai F, Wu J. “An extended localized algorithm for connected dominating set formation in Ad hoc wireless networks”. IEEE Trans Parallel Distrib Syst 2004;15(10):908–920.

De Couto DSJ, Aguayo D, Bicket J, Morris R. “A high-throughput path metric for multi-hop wireless routing”. Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom’03); 2003. pp. 134–146.

Ding G, Sahinoglu Z, Orlik P, Zhang J, Bhargava B. “Tree-based Data broadcast in IEEE 802.15.4 and ZigBee networks”. IEEE Trans Mobile Comput 2006;5(11):1561–1574.

Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. San Francisco (CA): Freeman; 1978.

Goel N, Kalaichelvan K, Karmouch E, Nayak A, Stojmenovic I, Villanueva-Pena E. “Physical layer impact on finding local knowledge information in Ad hoc and sensor networks”. Int J Comput Sci 2008;2(2):233–249.

Guha S, Khuller S. “Approximation algorithms for connected dominating sets”. Algorithmica 1998;20(4):374–387.

Heissenbüttel M, Braun T, Wälchli M, Bernoulli T. “Optimized stateless broadcasting in wireless multi-hop networks”. Proceedings of the 25th Conference on Computer Communications (Infocom 2006); 2006. pp. 1–12.

Heinzelman WR, Chandrakasan A, Balakrishnan H. “Energy-efficient communication protocol for wireless microsensor networks”. Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS); Volume 8; 2000. pp. 8020.

Hou J, Li N, Stojmenovic I. “Topology construction and maintenance in wireless sensor networks”. In: Stojmenovic I, editor. Handbook of sensor networks: algorithms and architectures: Wiley; 2005. pp. 311–341.

IEEE Standard 802. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs); 2003.

Ingelrest F, Simplot-Ryl D. "Localized broadcast incremental power protocol for wireless ad hoc networks". Wireless Netw 2008;14:309–319; Technical Report INRIA RT-0290, Jan 2004.

Ingelrest F, Simplot-Ryl D, Stojmenovic I. "Energy-efficient broadcasting in wireless Ad hoc networks". In: Cardei M, Cardei I, Du DZ, editors. Resource management in wireless networking: Springer; 2005. pp. 543–582.

Ingelrest F, Simplot-Ryl D, Stojmenovic I. "Routing and broadcasting in hybrid ad hoc and sensor networks". In: Wu J, editor. Theoretical and algorithmic aspects of sensor, Ad hoc wireless and peer-to-peer networks. Auerbach Publications (Taylor & Francis Group); 2006a. pp. 415–426.

Ingelrest F, Simplot-Ryl D, Stojmenovic I. "Optimal transmission radius for energy efficient broadcasting protocols in ad hoc and sensor networks". IEEE Trans Parallel Distrib Syst 2006b;17(6):536–547.

Ingelrest F, Simplot-Ryl D, Guo H, Stojmenovic I. "Broadcasting in ad hoc and sensor networks". Volume II, The handbook of computer networks, 3 volume set (Hossein Bidgoli, EIC): Wiley; 2007. Chapter 127.

Johnson D, Maltz D. "Dynamic source routing in Ad hoc wireless networks". In: Imielinski T, Korth H, editors. Mobile computing. Kluwer Academic Publishers; 1996. pp. 153–181.

Khabbazian M, Bhargava VK. "Localized broadcasting with guaranteed delivery and bounded transmission redundancy". IEEE Trans Comput 2008a;57(8):1072–1086.

Khabbazian M, Bhargava VK. "Efficient broadcasting in mobile Ad hoc networks". IEEE action on mobile computing 2008b;8(2):231–245.

Khan AA, Stojmenovic I, Zaguia N. "Parameterless broadcasting in static to highly mobile wireless Ad hoc, sensor and actuator networks". Proceedings. of the 22nd IEEE International Conference on Advanced Information Networking and Applications (AINA2008); 2008. pp. 620–627.

Kruskal JB. "On the shortest spanning subtree of a graph and the traveling salesman problem". Proc Am Math Soc 1956;7(1):48–50.

Li N, Hou C, Sha L. "Design and analysis of an MST-based topology control algorithm". Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003); 2003. pp. 1702–1712.

Li XY, Wang Y, Wan PJ, Frieder O. "Localized low weight graph and its applications in wireless Ad hoc networks". Proceedings of the 23rd Conference of the IEEE Computer and Communications Societies (INFOCOM 2004); 2004. pp. 431–442.

Lin CR, Gerla M. "Adaptive clustering for mobile wireless networks". IEEE J Sel Areas Commun 1997;15(7):1265–1275.

Liu H, Jia X, Wan P. "On energy efficiency in wireless Ad hoc networks". In: Xiao Y, Pan Y, editors. Volume 2, Ad hoc and sensor networks. Nova Science Publishers; 2005. 31–54.

Liu H, Jia X, Wan P, Liu X, Yao F. "A distributed and efficient flooding scheme using one-hop information in mobile Ad hoc networks". IEEE Trans Parallel Distrib Syst 2007;18(5):658–671.

Lim H, Kim C. "Multicast tree construction and flooding in wireless Ad hoc networks". Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'00); 2000. pp. 61–68. see also Comput Commun 2001;24(3–4): 353–363.

Lovasz L. "On the ratio of optimal integral and fractional covers". Discrete Math 1975;13:383–390.

Ma J, Gao M, Zhang Q, Ni LM. "Energy-efficient localized topology control algorithms in IEEE 802.15.4-based sensor networks". IEEE Trans Parallel Distrib Syst 2007;18(5):711–720.

McLaughlan B, Akkaya K. "Coverage-based clustering of wireless sensor and actor networks". Proceedings of IEEE International Conference on Pervasive Services IPCS; 2007. pp. 45–54.

Min M, Du H, Jia X, Huang CX, Huang SCH, Wu W. "Improving construction for connected dominating set with steiner tree in wireless sensor networks". J Glob Optim 2006;35:111–119.

Ni SY, Tseng YC, Chen YS, Sheu JP. “The broadcast storm problem in a mobile Ad hoc network”. Proceedings of the Fifth annual ACM/IEEE International Conference on Mobile Computing and Networking; 1999. pp. 151–162.

Ovalle-Martinez F, Nayak A, Stojmenovic I, Carle J, Simplot-Ryl D. “Area based beaconless reliable broadcasting in Ad hoc and sensor networks”. Int J Sens Netw (IJSNet) 2006;2(2):147–159.

Ovalle-Martinez FJ, Stojmenovic I, Garcia-Nocetti F, Solano-Gonzalez J. “Finding minimum transmission radii and constructing minimal spanning trees in Ad hoc and sensor networks”. Proceedings of the 3rd Workshop on Efficient and Experimental Algorithms, LNCS 3059; 2004. pp. 369–382.

Perkins C, Royer EM, Das SR. “Ad hoc on demand distance vector (AODV) routing”. Internet Draft, draft-ietf-manet-aodv-04.txt. 1999 Oct.

Peng W, Lu X. “On the reduction of broadcast redundancy in mobile Ad Hoc networks”. Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM MobiHoc); 2000. pp. 129–130.

Qayyum A, Viennot L, Laouiti A. “Multipoint relaying for flooding broadcast messages in mobile wireless networks”. Proceedings of the Hawaii International Conference on System Sciences (HICSS’02); 2002. pp. 3866–3875.

Shaikh JA, Solano-Gonzalez J, Stojmenovic I, Wu J. “New metrics for dominating set based energy efficient activity scheduling in ad hoc networks”. Proceedings of IEEE Conference on Local Computer Networks/WLN; 2003. pp. 726–735.

Simplot-Ryl D, Stojmenovic I, Wu J. “Energy efficient backbone construction, broadcasting, and area coverage in sensor networks”. In: Stojmenovic I, editor. Handbook of sensor networks: algorithms and architectures: Wiley; 2005. pp. 343–379.

Stojmenovic I. Comments and corrections to “dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks”. IEEE Trans Parallel Distrib Syst 2004;15(11):1054–1055.

Stojmenovic I. “Localized network layer protocols in sensor networks based on optimizing cost over progress ratio”. IEEE Netw 2006;20(1):21–27.

Stojmenovic I. “Energy conservation in sensor and sensor-actuator networks”. In: Wu S-L, Tseng Y-C, editors. Wireless Ad hoc networking: personal-area, local-area, and sensory-area networks: Auerbach Publications T&F; 2007. Chapter 4, pp. 107–133.

Stojmenovic I, Nayak A, Kuruvila J. “Design guideline for routing protocols in Ad hoc and sensor networks with a realistic physical layer”. IEEE Commun Mag 2005a;43(3):101–106.

Stojmenovic I, Nayak A, Kuruvila J, Ovalle-Martinez F, Villanueva-Pena E. “Physical layer impact on the design and performance of routing and broadcasting protocols in Ad hoc and sensor networks”. Comput Commun 2005b;28(10):1138–1151.

Stojmenovic I, Olariu S. “Data-centric protocols for wireless sensor networks”. In: Stojmenovic I, editor. Handbook of sensor networks: algorithms and architectures: Wiley; 2005. pp. 417–456.

Stojmenovic I, Seddigh M. “Broadcasting algorithms in wireless networks”. Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on Internet; 2000.

Stojmenovic I, Seddigh M, Zunic J. “Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks”. IEEE Trans Parallel Distrib Syst 2002;13(1):14–25.

Stojmenovic I, Wu J. “Broadcasting and activity scheduling in Ad hoc networks”. In: Basagni S, *et al.*, editors. Mobile Ad hoc networking: IEEE Press; 2004. pp. 205–229.

Toussaint G. “The relative neighborhood graph of a finite planar set”. Pattern Recognit 1980;12(4):261–268.

Viswanath K, Obraczka K. “An adaptive approach to group communications in multi-hop Ad hoc networks”. Proceedings of International Conference on Networking (ICN 2002); 2002. pp. 559–566.

Wan P-J, Alzoubi KM, Frieder O. “Distributed construction of connected dominating sets in wireless ad hoc networks”. Mobile Netw Appl (MONET) 2004;9(2):141–149.

Wang S-C, Wei DSL, Kuo S-Y. “SPT-based topology algorithm for constructing power efficient wireless ad hoc networks”. Proceedings of the ACM International World Wide Web Conference; 2004. pp. 234–235.

Wiese A, Kranakis E. “Impact of locality on location aware unit disk graphs”. Algorithms 2008;1(1):2–29.

Wieselthier J, Nguyen GD, Ephremides A. “On the construction of energy-efficient broadcast and multicast trees in wireless networks”. Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2000); 2000. pp. 585–594.

Wu J. “An enhanced approach to determine a small forward node set based on multipoint relay”. Proceedings of the IEEE 58th Vehicular Technology Conference (VCT 2003), Volume 4; 2003. pp. 2774–2777.

Wu J, Li H. “On calculating connected dominating set for efficient routing in Ad hoc wireless networks”. Proceedings of the 3rd International Workshop on Discrete Algorithm and Methods for Mobile Computing and Communications (DIAL-M); 1999. pp. 7–14.

Wu J, Dai F, Gao M, Stojmenovic I. “On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks”. IEEE/KICS J Commun Netw 2002;4(1):59–70.

Wu J, Wu B, Stojmenovic I. “Power-aware broadcasting and activity scheduling in Ad hoc wireless networks using connected dominating sets”. Wireless Commun Mobile Comput 2003;4(1):425–438.

Xia D, Vlajic N. “Near-optimal node clustering in wireless sensor networks for environment monitoring”. Proceedings of Canadian Conference on Electrical and Computer Engineering (CCECE’06); 2006. pp. 1825–1829.

Xu Y, Heidemann J, Estrin D. “Geography-informed energy conservation for Ad hoc networks”. Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom 2001); 2001. pp. 70–84.

Chapter 3

Sensor Area Coverage

Hai Liu[1], **Amiya Nayak**[2], **and Ivan Stojmenovic**[2]

[1]*Hong Kong Baptist University, Hong Kong, P.R. China*
[2]*School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5*

Abstract

Sensor networks normally have redundancy for sensing coverage. Some sensors are allowed to sleep while preserving network functionality. Sensors that are randomly placed in an area should decide which of them should be active and monitor an area, and which of them may sleep and become active at a later time. The connectivity is important so that the measured data can be reported to a monitoring center. Sensor area coverage problem has been considered for both the unit disk graph and physical layer-based sensing models. Actuators may similarly run a protocol to decide about their service areas, releasing some of them from particular duty. Operational range assignment for both sensor and actuator nodes is also discussed.

3.1 PROBLEMS, MODELS, AND ASSUMPTIONS

One of the fundamental issues in wireless sensor networks (WSNs) is the sensor coverage problem. Sensor coverage is to deploy a set of sensor nodes in an area of interest for monitoring and/or tracking. Sensor nodes are normally densely deployed in WSNs. To prolong the network lifetime, sensors should sleep as much as possible. Ideally, they should wake up only when they are really needed. However, this may not be possible since additional hardware may be required for such ability. For example, a radio-triggered hardware component was introduced by Gu and Stankovic (2004). Since the events of interest often contain energy, their energy can be used to trigger the added hardware component which then in turn initiates the transition of the system from sleep mode to wake-up mode. Existing sensors, however, are not equipped with such hardware. In these cases, when sensors decide to enter sleep mode, they set their clock for waking at a predetermined time, regardless of events nearby. Wireless sensor networks and wireless sensor actuator networks (WSANs) employ collaborative mechanisms for scheduling wake-up and sleep periods.

Wake-up and sleep periods exist for both sensing and communication hardware components. Normally, a portion of nodes is required to be active (with respect to certain hardware) to perform the given tasks, while other nodes could sleep to save energy. Both sensor and actuator networks have node redundancies for communication and/or sensing, since only some nodes are needed for traffic forwarding, monitoring, or servicing. *Activity scheduling* is to decide which nodes should be active and which may be allowed to sleep. The decisions are periodically reevaluated, and the problem is also known as *duty cycling*. There are different levels of activity. Sensor nodes may turn off both sensing and communication hardware and therefore fully be in a sleep mode. A set of sensor nodes that together fully cover a given area is frequently referred to as the *area-dominating set*, or as the *sensing backbone* for the network. The *communication backbone* is normally built on top of the sensing backbone. That is, some of the sensors may have an active sensing device but a passive transmitter and receiver hardware. Their communication needs can be fulfilled by their neighbors in the communication backbone (neighbors whose communication hardware is always turned on). We have discussed communication backbone construction techniques in the previous chapter. While this chapter refers to the sensor area coverage problem, actuators can similarly be considered for the analogous actuator coverage problem, deciding which of them are needed to service sensors in their areas while allowing others to rest or perform other duties. This problem occurs when or if the actuator network is dense and has redundancies.

The chapter focuses on the design of the wake-up or sleep schemes for area coverage problems in WSNs and WSANs. In a typical area coverage problem, a set of sensors is distributed over a given area. Each sensor is able to cover a small area, which is normally assumed to be a circle with radius centered at it. The problem is to find a subset of sensors that are connected and still cover the same area, such that these sensors alone are able to perform the monitoring task.

Full coverage, maximum network lifetime, and connectivity are critical requirements of any area coverage protocol. There are a variety of problem statements, assumptions, and solution approaches for the problem of sensor area coverage. The chapter focuses on the area coverage problem in which each point in a given geographic area should be covered by at least one sensor. The main objective of area coverage protocol is to achieve full area coverage by a subset of sensors with the minimal possible number of sensors in the subset.

Assumptions about sensing radii (SR) may vary. In most articles, the SR of nodes is assumed to be fixed and the same for all sensors (Tian and Georganas, 2002). A more general case is when SR are fixed for each sensor, but are not the same. SR is assumed to be adjustable in some articles (Wu and Yang, 2004).

The area coverage problem can be generalized by requesting multiple coverage for each point in the area. The most straightforward generalization is k-coverage problem. An area is k-covered if every point of the area is covered by at least k distinct sensors. A more restrictive generalization is k-layer coverage problem, which requires k disjoint subsets of sensors so that each of these k subsets provides one-coverage (fully covers the area). A k-layer coverage is also a k-cover, but the converse may not hold. In the example in Figure 3.1, the shaded area centered at O is two-covered since every point in the area is covered by at least two circles. However, we can not find two distinct subsets of sensors such that each one fully covers the area.

Sensing and communication are normally modeled as unit disk graphs (UDGs), with the corresponding sensing radius (SR) and communication radius (CR) denoted by these terms in the rest of the chapter. In the UDG model, a sensor is able to monitor location of an event if and only if the location is within the SR of the sensor. In reality, sensing ability decreases with distance, which can be exploited in a physical layer model, where the probability of sensing an event depends on the distance from the sensor to the event. Two nodes are communication neighbors if they are within distance CR from each other. Sensing neighbors are two nodes whose corresponding sensing areas overlap. If their sensing areas are disks, then they are sensing neighbors if the distance between them is less than the sum of their corresponding sensing ranges.

The sensor network may operate with or without time synchronization among sensor nodes. In a *synchronous* protocol, all sensor nodes maintain a common clock by applying some synchronization protocols (Li and Rus, 2004; Romer *et al.*, 2005). The sensor nodes coordinate with each other to make their activity schedules according to the common clock. All decisions are made in rounds. That is, all nodes could wake up at the same time, exchange messages, and then decide which of them will be active. The ZigBee standard requires sensor

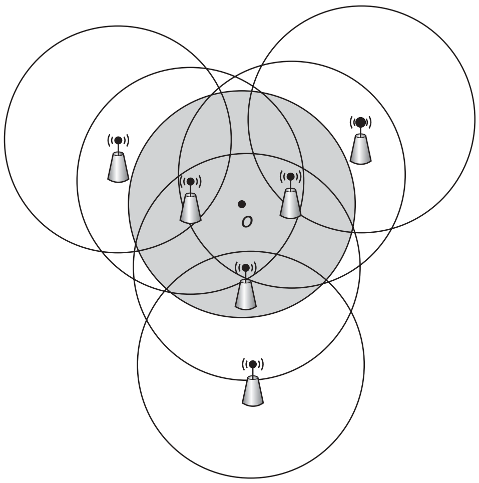


Figure 3.1 The shaded area is two-covered but not two-layer covered.

nodes to be time-synchronized. Synchronous behavior provides advantages for energy-efficient communication in addition to efficient area coverage protocols.

In an *asynchronous* protocol, sensor nodes do not follow a common clock. Each node makes its own decision to be active or to sleep for a period of time, based on its individual clock. Each node in an asynchronous protocol may wake up at its predetermined time and decide whether it needs to be active, based on a message exchanged with currently active neighbors.

It has been proven that finding the minimum number of connected working nodes that cover the area of interest is NP-hard (Kumar *et al.*, 2000; Gupta *et al.*, 2003). Since it is NP-hard for even centralized algorithms, finding localized algorithms to achieve good performance is an even more challenging task.

3.2 COVERAGE AND CONNECTIVITY CRITERIA

Coverage and connectivity criteria serve as building blocks in area coverage algorithms by providing computationally efficient ingredients. The choice of criteria depends on the ratios of SR and transmission radius and their uniformity.

If the SR and transmission radius are equal, the coverage property can be tested by verifying whether or not the whole perimeter of the sensing circle is covered by other circles. In the example in Figure 3.2a, sensing area of node O is not fully covered by two other circles since there are two uncovered segments. The correctness of the criterion follows from the following observation: If two sensing circles intersect, then each center is inside the other circle. If point A on the perimeter of circle O is covered by a circle centered at P, all points on the line segment from A to center O are covered by the same circle since both O and A are inside the circle centered at P and the sensing area of any circle is convex. Therefore, instead of testing the whole line segment OA for inclusion in an other circle, only endpoint A on the circle perimeter needs to be checked. Note that this (and the following) criteria assume that no two sensors are placed at the same location.

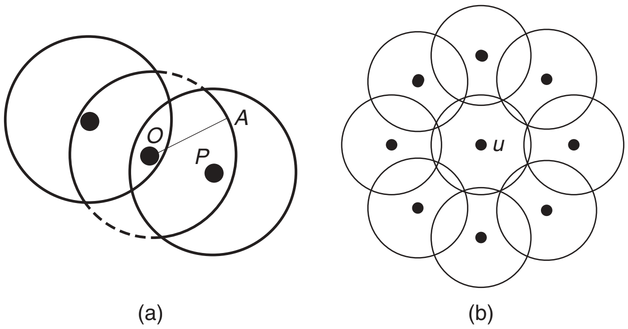


Figure 3.2 Perimeter-based coverage test.

This criterion can be generalized to the case of k-coverage. The monitoring region is k-covered if and only if the perimeter of each sensor is covered by at least k distinct sensors (Huang and Tseng, 2003). This criterion is not applicable if the CR is larger than the SR. In the example in Figure 3.2b (for $k = 1$), suppose that all nodes are within CR of node u. Although the perimeter of circle u is covered by other circles, sensing area of u is not fully covered.

Wang *et al.* (2003) and Zhang and Hou (2005) introduced a covering criterion to decide whether or not a sensing area is fully covered by other sensing areas. It does not require the uniform SR of nodes. Furthermore, it does not depend on ratio of the CR and the SR. It also generalizes to sensing areas of arbitrary shape (not just disks) and is applied on the corresponding boundaries.

Theorem 3.1. Coverage of circle centered at O by circles centered at $C_1, \ldots, C_m$ can be reduced to coverage of intersection points of two covering circles C_i and C_j, or of O and one covering circle C_i, that is inside the sensing area of O, as follows: If there are at least two covering circles and any such intersection point is covered by a distinct covering circle C_k, then the sensing area is fully covered.

Proof. The basic idea of proof (as given in Gallais *et al.* (2008)) is illustrated in Figure 3.3a. Suppose that there is a point P, which is not covered by any sensor in the region. P lies in an uncovered patch that is bounded by only exterior arcs of a collection of sensing circles and/or boundary of the sensing area. In Figure 3.3a, the uncovered patch is the shadow area Q-R-S-T-U. Travel (in any direction) from P to the boundary of the uncovered patch and follow the boundary until it meets an intersection point (e.g., point Q in Fig. 3.3a) of two covering circles C and D (or intersection point U of covering circle D and central circle O) on the boundary. Q is not covered by any third covering circle. It contradicts the condition of the theorem. ♦

The central gray circle in Figure 3.3b is fully covered since any intersection point of two circles inside the gray area is covered by a third circle. For instance, intersection point P of circle A and circle C is covered by circle B, and intersection point T of circle A and circle D is covered by circle C.

The criterion in Theorem 3.1 provides an efficient method for testing full coverage of a sensing area. However, it does not provide direct information about the possible size of the uncovered region if the region is not fully covered. One possible estimate is to randomly generate a certain number of points and test coverage of each point with existing circles. The uncovered region could be estimated by the percentage of uncovered points. The alternative is to make an estimate based on the area of the polygon with same vertices (e.g., polygon $QRSTU$ in Fig. 3.3a). There exists a need for designing more accurate and fast-coverage size-estimation protocols.

Theorem 3.1 can be extended to the three-dimensional scenarios. In Ovalle-Martinez *et al.* (2006), it is used for the broadcasting protocol in 3D, where

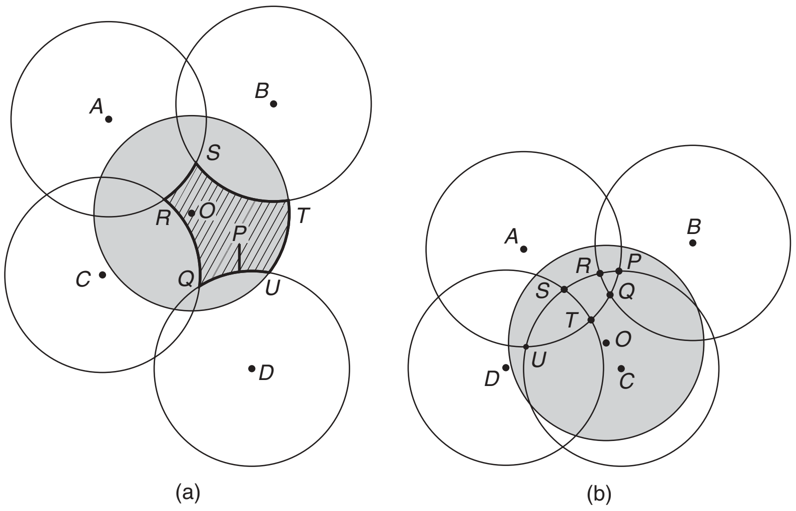


Figure 3.3 The gray circle is fully covered if all intersection points are covered.

each node has the same transmission radius. This corresponds to the scenario for sensor volume coverage where CR = SR (details are in Chapter 2). In this case, the coverage criterion can be expressed as follows:

Theorem 3.2. Ovalle-Martinez *et al.* (2006) Suppose that sphere A is intersected by spheres $C_1, C_2, \ldots, C_m$. Consider all intersection points X on the 3D perimeters of the spheres centered at A, C_i, and C_j. If there exists at least one such intersection point and every such intersection point X is located inside at least one of the remaining spheres (centered at C_k for some k) then the sphere centered A is fully covered by the spheres centered at $C_1, C_2, \ldots, C_m$.

In case of arbitrary ratios of CR/SR, and sensing volumes of arbitrary shapes (for simplicity, only spherical volumes are stated in the criterion), the following generalization holds. Again, no two sensors are assumed to be in the same location.

Theorem 3.3. Suppose that sphere A is intersected by spheres $C_1, C_2, \ldots, C_m$. Consider all intersection points X of spheres C_i, C_j, C_k (or by C_i, C_j, and A) located inside sphere A. If there exists at least one such intersection point and every such intersection point X is located inside at least of one of the remaining spheres (centered at C_p for some$p$) then the sphere centered A is fully covered by the spheres centered at $C_1, C_2, \ldots, C_m$.

Proof. The proof of Theorem 3.3 is similar to the proof of Theorem 3.1. Suppose that the volume centered at A is not fully covered by other volumes. Let P be one of uncovered points. It is located inside a 3D uncovered patch. "Travel" from P

until a boundary is met. It could be boundary of one of the covering circles, or boundary of *A*. Traverse further that boundary until a line of intersection of two spheres is met. Traverse that line until a point of intersection with a third sphere is encountered. This point is not covered (it is not inside) in any other sphere, which contradicts the assumptions in the theorem. ♦

Both the coverage and connectivity criteria were studied in Zhang and Hou (2005) and Wang *et al.* (2003). It is proved that if the transmission radius is at least twice the SR and the area to be covered is convex, then the area coverage also implies connectivity of the covering sensors. Any two nodes whose sensing circles intersect are then neighbors within Tian (2004) generalized the proof by eliminating the convexity condition.

Theorem 3.4. If the transmission radius is at least twice the SR then the area coverage also implies connectivity of the covering sensors.

Proof. The proof (Tian, 2004) is as follows: If a network is not connected then there are at least two connected components in the network. The distance of any pair of nodes that are selected from two components, respectively, is larger than CR. Since CR $>$ 2SR, there is no intersection between coverage area of nodes in two components. Therefore, the whole region is not fully covered since the region is continuous (Tian, 2004). ♦

The relationship between the degree of coverage and connectivity was further studied in Wang *et al.* (2003). A graph is k-connected if it remains connected when any $k - 1$ vertices are deleted from the graph.

Theorem 3.5. Wang *et al.* (2003) A set of nodes that k-cover a convex region forms a k-connected communication graph if the communication radius is at least twice the sensing radius (CR $>$ 2SR).

Several centralized sensor area coverage algorithms are surveyed in Simplot-Ryl *et al.* (2005). Since centralized algorithms are inefficient in gathering information, due to their communication overhead, they are suitable only for very small networks. We will discuss here only localized algorithms. The presented algorithms all assume that each sensor is aware of its own location (geographic coordinates).

3.3 AREA-DOMINATING SET BASED SENSOR AREA COVERAGE ALGORITHM

Sheu *et al.* (2007) proposed a localized protocol to find a set of connected sensor nodes to cover the required region in heterogeneous sensor networks. Sensor nodes may have different SR and CR. A sensor may need multiple hops to reach its sensing neighbors if SR $>$ CR. This case is of theoretical interest only since in practice CR > SR. The protocol consists of three phases: *neighbor discovery*, *self-pruning*, and *active sensing neighbors discovery*.

Each sensor collects information on its sensing neighbors by "hello" messages (neighbor discovery). The node information includes a node's ID, sensing range, location, and priority. The priority could be residual energy, sensing range, or communication degree, or a combination of several metrics (priorities are assumed to be distinct among nodes). Note that flooding is required for a node to learn its sensing neighbors if SR > CR. Only this phase requires message exchanges among nodes. The remaining two are decisions made by each node without communicating with others.

In the self-pruning phase, each node determines whether or not to be active. It decides to be active if its sensing area is not completely covered by the union of sensing areas of its sensing neighbors that have higher priority than the node. After this phase, the required region is fully covered by the active sensing nodes.

In the active sensing neighbor discovery phase, each sensing node A determines active sensing neighbors. Several sensing neighbors may cover the same segment of A's perimeter. A recognizes a sensing neighbor B as active if a segment of its perimeter is covered by B and B has the highest priority among all sensing neighbors that cover the same segment of the perimeter.

Suppose the sensing range is the priority and a node with a larger sensing range has a larger priority value. In the example in Figure 3.4, suppose that the order of priorities of nodes is $A < B < C < D < E < F < G$. In the *self-pruning* phase, nodes A, B, D, E, F, G decide to be active since their sensing area is not fully covered by neighbors with higher priority. Node B is active since priority of A is lower than B, while node C is passive since priority of both F and G is higher than C. In *active sensing neighbors discovery* phase, the perimeter of G is divided into segments by its sensing neighbors. Since segments S_1, S_4, and S_5 are covered only by nodes A, D, and F respectively, A, D, and F are recognized as active sensing neighbors by G. Segment S_2 is covered by both A and B, and B is recognized as an active sensing neighbor by G since priority of B is higher than the priority of A. Similarly, segment S_3 is covered by B and D, but it is also covered by E which has a higher priority than both B and D. So, E is also recognized as an active sensing neighbor by G.

Note that a node decides to be passive based on coverage by higher priority sensing neighbors, and some of them might also decide to be passive. However, in doing so, these neighbors also recognized higher priority covering neighbors. For each patch of area to be covered, this leads to a chain of higher priority sensing neighbors, and the one with the highest priority must be active. Thus, the set of active sensors indeed covers the original area. The connectivity follows from the originally constructed paths to each sensing neighbor, including those recognized to be active. The proposed protocol could be applied in both synchronous and asynchronous networks since only neighbor discovery phase involves message exchanges. Decisions can be made originally in synchronous mode following "hello" message exchanges. Later, dynamic changes in the network can be handled locally and asynchronously, as they are occurring.

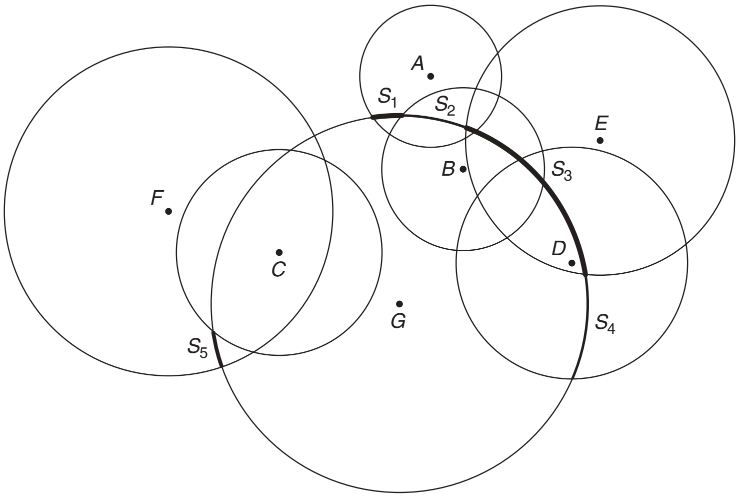


Figure 3.4 Area-dominating set based coverage.

3.4 ASYNCHRONOUS SENSOR AREA COVERAGE

3.4.1 PEAS

Ye *et al.* (2003) proposed probing environment and adaptive sleeping (PEAS), which is a localized threshold-based protocol for dynamically selecting an area-covering set in asynchronous sensor networks. Each sensor is assumed to have the same probing radius p and the same maximum CR, which is also the maximum SR. Initially, all nodes are sleeping and the sleep duration is an exponentially distributed random number. The protocol consists of two phases, probing environment and adaptive sleeping.

When a node A wakes up, it broadcasts a probing message using probing radius p. Any active node receiving the message (that is, active neighbor of A at distance up to p) will send back a reply message to the node. Node A decides to work continuously if it does not hear any reply message from neighbors (this means that there is no active neighbor within distance p of A). Once a sensor decides to be active, it continues to work until it depletes its energy. Otherwise, the node selects a new sleeping duration and goes back to sleep mode. It will wake up at a later predetermined time to reevaluate the decision. In adaptive sleeping phase, each active sensor measures the current accumulated wake-up rates of its sleeping neighbors. The measured rate is included in the reply message, which will be sent back to any probing neighbor. The probing node then adjusts its sleeping time accordingly, with the goal of having a relatively constant wake-up rate. The density of active nodes can be controlled by parameter p.

Consider the example in Figure 3.5. The circles with smaller radii are the probing areas while the circles with larger radii are the communication areas.

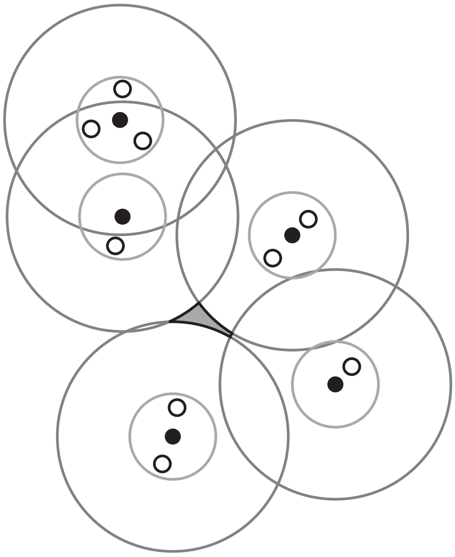


Figure 3.5 PEAS protocol for sensor area coverage.

Black nodes are active while white nodes are in sleep mode. PEAS cannot guarantee full coverage of the monitored area. For example, shaded area between circles in Figure 3.5 is covered by sleeping white nodes but not by corresponding active black nodes located within distance p. The probability of full coverage is shown to be close to 1 if the threshold p is less than $1/(1+\sqrt{5}) \approx 0.3$ of the SR, that is, $p_{\mathrm{r}} < 0.3\mathrm{SR}$. Activation of more sensors by a smaller probing radius has an insufficient contribution toward covering some new area. The proposed protocol has a high degree of fault tolerance. In a modification of PEAS (Gui and Mohapatra, 2004), active nodes report their activity time left, so that sleeping nodes can adjust their future wake-up times.

Gui and Mohapatra (2004) proposed a simple sleeping scheme for sensor activity scheduling. Each node independently follows its own sleep schedule without collaborating with each other on the sleeping issues. To achieve the required quality of surveillance, the probability p of selecting active status by any node is determined by n/N, where n is the required number of active sensors and N is the total number of sensors. Therefore knowledge of n and N, and uniform node distribution is also desirable for good performance.

3.4.2 ACOS

Cai *et al.* (2007) proposed an asynchronous area-based collaborative sleeping (ACOS) protocol for WSNs. The problem is to schedule states of sensors to maximize the coverage and minimize the energy consumption. The ACOS protocol cannot guarantee full coverage of the sensing area. It is based on calculation of the *net sensing region* of a sensor, which is the region in the sensing range of

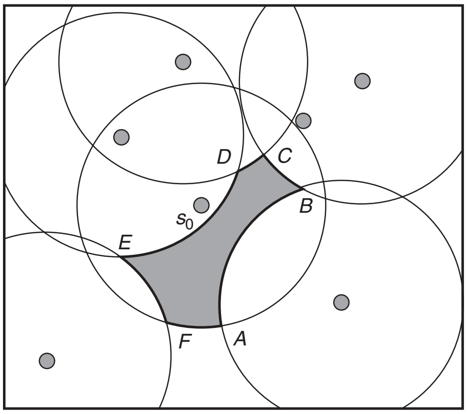


Figure 3.6 The net sensing region for sensor s_0.

the node that is not in the sensing range of any other active sensor. The definition is illustrated in Figure 3.6. The shaded area is the net sensing region of s_0.

The algorithm proposed in Huang and Tseng (2003) is employed to compute boundary of the net sensing region. In this algorithm, sensor s_0 computes the portions of perimeter of each of its covering circles, which is not covered by other covering circles, and a similar portion of its own perimeter. This generates a list of uncovered arcs, which can be then linked into a list. In the example in Figure 3.6, arcs AB, ED, BC, EF, AF, and CD are identified first, leading then to a circular list $ABCDEF$ (note that there may be more than one uncovered region). The area of each sensing region can be approximated by the area of the polygon formed by its segment sequence. In the example in Figure 3.6, the area of the net sensing region of s_0 is approximated with the polygon $ABCDEF$.

When a node wakes up, it broadcasts a message to its neighbors within 2SR and waits for a time-out period. Upon receiving the message, each active neighbor replies with a message, which includes its location and the remaining awake time. When the time-out expires, the node can compute the *net area ratio* based on information it receives from awake neighbors. The net area ratio of a node is defined by the area of the net sensing region divided by the area of the maximum sensing region of the node. If the ratio is less than a predetermined threshold, the node returns to sleep state and sleeps during the minimum awake times of its neighbors. If the ratio is equal to or greater than the threshold, the node changes to awake state and decides how long to be awake. The decision is then transmitted to its neighbors.

3.5 SYNCHRONOUS SENSOR AREA COVERAGE

3.5.1 Coverage with Low Communication Overhead

Tian and Georganas (2002) proposed a localized coverage-preserving node scheduling algorithm for synchronous sensor networks. The algorithm guarantees full coverage of the sensing area. A node decides to go to sleep mode if and only if the sensing area of the node is fully covered by the union of sensing areas of its active neighbors. A random back-off scheme was proposed to determine the time nodes make decisions. Decisions to turn off are announced to neighbors. Upon receiving such an announcement, adjacent nodes with longer back-off time periods will delete the sender's information from active neighbor lists. Therefore, sleeping neighbors (at decision time) will not be considered for sensing coverage. The algorithm was extended by Jiang and Dou (2004) by adding a round of location information exchange, and applying the perimeter coverage criterion.

Gallais *et al.* (2008) proposed several localized and synchronous sensor area coverage protocols for heterogeneous sensor networks. Each sensor may have its own SR and transmission radius (Gallais *et al.*, 2008). The covering criterion presented in Theorem 3.1 was used to efficiently decide whether or not a sensing area is fully covered by neighboring sensors. Neighbor discovery is not needed. Nodes wait for a random time-out duration while receiving activity decision messages from neighbors. The maximum time-out duration can be tailored to prefer sensors that were not elected in previous rounds, or sensors with more remaining energy. Each sensor evaluates its coverage and connectivity by active neighbors and decides whether or not to be active when its time-out expires. Active sensors inform neighbors, whereas decisions to sleep may or may not be announced (by withdrawal messages). After making a decision to be active, nodes may hear from more active neighbors, and their sensing area may then become fully covered. Such nodes may then change their minds by sending a retreat message to their neighbors. On the basis of whether or not the *withdrawal* message and the *retreat* message are transmitted, four variants of the protocol were proposed in Gallais *et al.* (2008).

In the activity, withdrawal, and retreat (AWR) variant, for example, each message is transmitted if its corresponding condition is satisfied. In the example in Figure 3.7, CR = SR, and each node selects a random time-out and waits for messages from neighbors. Suppose that $timeout_1 < timeout_2 < timeout_3 < timeout_4 < timeout_5 < timeout_6$. Nodes 1, 2, and 3 decide to be active after their time-out expires since their sensing areas are not fully covered. They send an *active* message to their neighbors. In Figure 3.7a, the gray region is covered by active nodes 1, 2, and 3. Then node 4 decides to be active and sends an *active* message to nodes 1, 3, 5, and 6. Since node 5 receives the *active* message from nodes 1, 2, and 4, it decides to go to sleep mode and sends a *withdrawal* message to nodes 1, 2, and 4. Afterwards, node 6 decides to be active. After receiving the *active* message from node 6, node 4 finds that its sensing area is fully covered by nodes 1, 3, and 6 (Fig. 3.7b). Thus, node 4 changes its decision to go to sleep and sends a *retreat* message to nodes 1, 3, 5, and 6. Finally, nodes 1, 2, 3, and 6 are active while nodes 4 and 5 decide to sleep.

Experimental results with the ideal medium access layer (MAC) layer show that for a similar number of selected active sensors, four methods (Gallais *et al.*, 2008) significantly reduce the number of messages to decide the status

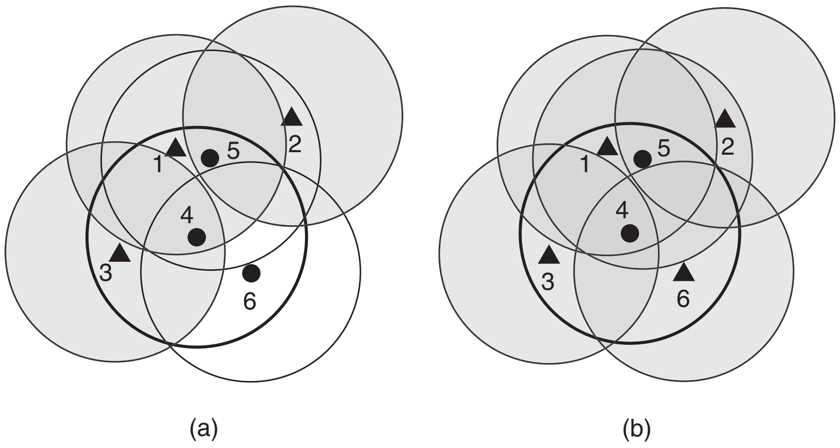


Figure 3.7 Activity, withdrawal, and retreat variant of sensor area coverage algorithm.

compared to localized protocol (Tian and Georganas, 2002; Jiang and Dou, 2004) where nodes send "hello" messages followed by retreat messages before sleeping. Message losses were also considered (Gallais *et al.*, 2008), induced by a MAC layer with collisions and/or a realistic physical layer, and showed that the existing compared method (Tian and Georganas, 2002; Jiang and Dou, 2004) for dense networks fails to cover the area reasonably with a connected set of active nodes (nodes may decide to sleep since some retreat messages are not received, creating coverage holes, and connectivity losses). Methods (Gallais *et al.*, 2008), however, still remain robust in terms of high area coverage with a reasonable amount of active nodes and connectivity preservation despite message losses.

Gallais and Carle (2008) proposed a surface coverage relay protocol based on the backbone concept in Adjih *et al.* (2005). Each sensor node *A* first decides about its coverage relay sensing neighbors. It sorts its sensing neighbors by their decreasing distance, and they are considered as possible relays in that order. Thus, the furthest sensing neighbor of *A* is its relay. Candidate neighbor becomes a relay node if it covers a portion of sensing area of *A* not covered by previous relays. Relay sets of each sensor can be constructed after a "hello" message from each sensor, to learn its sensing neighbors. The same message may also be used to construct the connected dominating set with the MPR-DS algorithm (Adjih *et al.*, 2005) (described in Chapter 2), if CR is assumed to be equal to SR. This ensures that two-hop communication neighbors are also sensing neighbors. To construct active sensors for area coverage, the same rule from Adjih *et al.* (2005) can be applied. Every sensor whose key is lowest among the (communication) neighbors or which belongs to the relay set of the neighbor with the lowest key, must be active. The set of active nodes covers an area as large as the area covered by all sensors (Gallais and Carle, 2008).

3.5.2 Location and Calculation-Free Sleep Scheduling

Tian and Georganas (2004) studied the location and calculation-free node scheduling for large-scale WSNs. Three algorithms were proposed. In the *nearest neighbor-based node scheduling scheme*, each node is assumed to know the distance to its neighbors but not their locations. At the beginning of the scheme, each node transmits a short message to announce its existence to its neighbors. Each node generates a random back-off time and listens to the channel during the time. If it receives the turnoff decision message from a neighbor, it deletes the neighbor from the active neighbor list. Once the back-off time expires, the node decides its status. In the nearest neighbor-based node scheduling scheme, each node determines if the distance to its nearest active neighbor is less than or equal to the threshold D. If so, the node decides to go to sleep mode, and transmits the decision to its neighbors. Otherwise, the node decides to be active. The *neighbor number-based node scheduling scheme* also employs the random back-off procedure. After back-off time expires, each node counts the number of active neighbors. If the number is less than a predetermined threshold, the node decides to be active. Otherwise, it goes into the sleep mode, and transmits the decision to its neighbors. The threshold is determined to achieve the required percentage of uncovered sensing area of individual nodes on an average.

The *probability-based node scheduling scheme* is based on a probability model and does not use random back-off procedure. In the scheme, each node generates a random number in (0, 1) and checks if the number is less than the predetermined threshold. If this is the case, the node decides to go to sleep mode; otherwise, it decides to be active. The three proposed schemes have low scheduling and time complexities. The disadvantage is that all these schemes cannot guarantee to preserve the original full network coverage, and are based on some parameters.

3.6 MULTICOVERAGE BY SENSORS

Fault tolerance of some applications may require that any point in the sensing area is covered multiple times. Even if single coverage suffices, it may be energy-efficient to partition and schedule the sensors to work in round-robin manner, such that the sensing area could be covered multiple times.

Gallais and Carle (2007) proposed an adaptive localized algorithm for multiple sensor area coverage. The algorithm follows the assumptions and algorithms, including options, outlined in Gallais *et al.* (2008) and in Section 3.5.1. The only difference is in the evaluation criterion applied to reach a decision. To address the k-coverage problem, the basic idea is to divide the sensing area into grids, and it is required that every grid point be covered by at least k neighbors. Alternatively, the k-cover criteria from Theorem 3.5 could be utilized.

In the k-layer coverage problem, active neighbors add their layer number to their positive acknowledgments. A node determines that its sensing area is fully k-layer covered if and only if there are at least k active layers and each of them fully covers sensing area of the node. The problem was studied in Simplot-Ryl *et al.* (2005). The algorithm (Simplot-Ryl *et al.*, 2005) adjusts k dynamically to reflect the sensor density. Each sensor node selects a time-out. Suppose that node A received a message from a neighbor that informed about i, the cover layer number selected by that neighbor, and its geographic coordinates. Node A adjusts its uncovered portion of layer i, and may adjust its time-out appropriately. When the time-out expires, a decision is made and transmitted. Node A assigns the layer j, which is a minimal number so that the area in layer j is not yet fully covered. Alternatively, among layers covered partially by some neighbors and not yet fully covered, A chooses one that maximizes the uncovered area. Another option is that, if all layers covered by some neighbors are fully covered, A may choose a new layer and inform neighbors about covering it.

3.7 PHYSICAL LAYER-BASED SENSING, PROTOCOLS, AND CASE STUDIES

Most of existing works on sensor area coverage employ the UDG model. The sensing area of a node is a disk with SR centered at the node itself. A point in the monitoring region is covered by a sensor if and only if, the point is inside the coverage disk of the sensor. However, as discussed in previous chapters, the UDG model is not realistic since variations of received signal strengths are not considered. Nondeterministic radio fluctuations cannot be ignored in sensor area coverage, and the monitoring ability of sensors is probabilistic.

Gallais *et al.* (2006) studied k-layer coverage in WSNs with realistic physical layers. The probability of detecting an event depends on the distance of sensor from it. The approximated function that is introduced in Kuruvila *et al.* (2004)is applied:

$$p(x) = \begin{cases} 1 - (x/\mathrm{SR})^{2\alpha}/2 & \text{if } 0 < x \leq \mathrm{SR}, \\ ((2\mathrm{SR} - x)/\mathrm{SR})^{2\alpha}/2 & \text{if } \mathrm{SR} < x \leq 2\mathrm{SR}, \\ 0 & \text{otherwise}, \end{cases} \tag{3.1}$$

where x is the distance between the point and the sensor, and α is the signal decline factor which depends on the environment (Eq. 3.1). The SR is selected so that the probability for a sensor to correctly sense a target, which is located at distance SR from the sensor is 0.5. The sensing function is drawn in Figure 3.8 where $\mathrm{SR} = 1$ and $\alpha = 2$. It resembles the lognormal shadowing model used to model communication between nodes.

The proposed sensor area k-layer coverage protocol (Gallais *et al.*, 2006) works as follows. Suppose sensors are time synchronized and work in a round-robin manner. Each node selects a waiting time-out, listening to messages from neighbors. Activity messages include the layer at which a node has decided to be active. Depending on the physical layer used for sensing modeling, any node can evaluate if the provided coverage is sufficient for each layer. If so, the node can sleep, otherwise it selects a layer to be active.

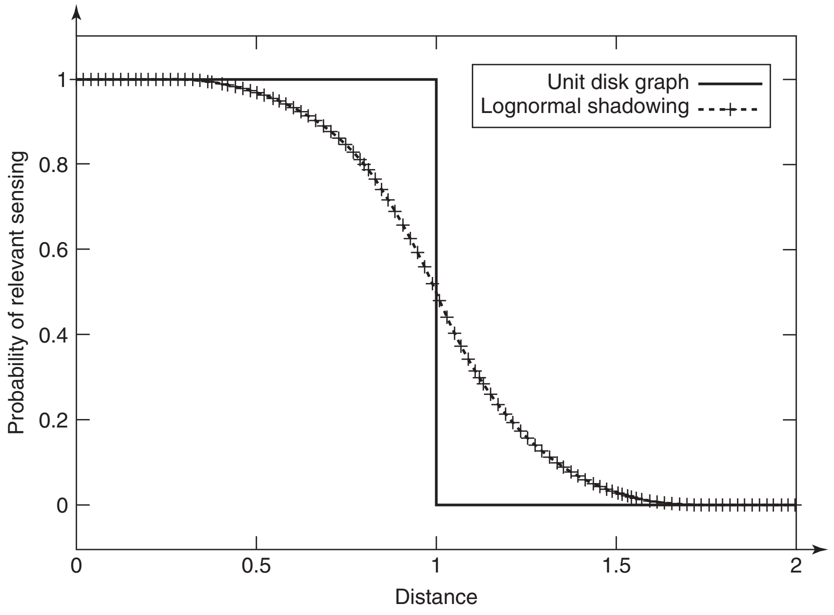


Figure 3.8 Lognormal shadowing model for sensing, SR $= 1$ and $\alpha = 2$.

To evaluate if the sensing area of a node is sufficiently covered by sensors at a given layer, the node, say u, randomly selects a set of physical points (inside the sensing circle with radius SR), denoted by S, and checks the coverage of these points. The number of selected points depends on the desired accuracy. For each point P in S, node u computes the probability that P could be correctly sensed by at least one of its neighbors. The coverage probability can be calculated as follows: $1 - \Pi_{i=1}^{|N(u)|}(1 - P(d_i))$, where $N(u)$ is the set of neighbors of u and d_i is distance between point P and neighbor i. For example, suppose that node u has three neighbors within its SR: $u1$, $u2$, and $u3$. Distances between point P and $u1$, $u2$, and $u3$ are $d1$, $d2$, and $d3$, respectively. Then the probability that P is correctly sensed by neighbors of u is $1-(1-P(d1))(1-P(d2))(1-P(d3))$.

Once the coverage probability of each point in S is computed, the average of all probabilities is used to compare with the predetermined threshold. If the average probability is greater than the threshold, the node believes its sensing area is covered.

3.8 OPERATION RANGE ASSIGNMENT IN WSANs

Younis *et al.* (2008) studied operation range assignment for both sensor nodes and actuator nodes in WSANs. A WSAN is randomly deployed in the field. It assumes that each node, either sensor or actuator, has k different operational ranges, where sensing range of sensors and acting range of actuators are referred to as operational ranges. Each node may not know the location information of its neighbors and itself. Each node is able to compute the relative positions of its neighbors (virtual coordinates) based on distance and two-hop information. The *operational range assignment protocol* (ORAP) (Younis *et al.*, 2008) is for asynchronous coverage. The goal is to assign operational range for each node, such that the field area is covered with a probability that is larger than a predetermined threshold, and the operational lifetime of each individual node is prolonged as much as possible. It is assumed that coverage probability decreases with distance and increases with selected operational range.

The basic idea of ORAP is to assign longer ranges to nodes that have more residual energy. Furthermore, ORAP is periodically retriggered to reassign the operational ranges, such that the load of nodes is balanced. Operational range assignment protocol works as follows. All nodes are undecided at initialization stage. Each node selects its *weight* as the ratio of residual and maximal energies. Suppose possible ranges of a node are $R_1 < R_2 < \cdots < R_k$. A node decides to use maximal range if its neighbors cannot cover its area with their maximal ranges. Otherwise, the node computes its operational range based on its weight and weights of its neighbors. A node v does not make a decision regarding its working range R unless the node has the highest weight among all its undecided neighbors. Node v first sets its range R to R_{k-1} and range of every undecided neighbor u to the largest R_j that is smaller than $[weight(u)/weight(v)]^{1/m} \times R$, $j \leq k - 1$, where energy consumption of the sensing or acting stage is assumed to be proportional to R^m. If this assignment covers the operational range of v, v reduces its range to R_{k-2}. The process repeats until node v at its particular range R_i cannot cover it. Node v then decides to use range R_{i+1}, changes its state to decided and informs its neighbors. Finally, nodes check for redundancies based on all finalized decisions. Node v waits to receive tokens from all the neighbors with less weight than its own. Once these tokens are available, v computes its final operational range and releases a token that advertises the new range of v. If the timer expires before the node receives enough tokens, it keeps its selected range.

REFERENCES

Adjih C, Jacquet P, Viennot L. “Computing connected dominating sets with multipoint relays”. Ad Hoc Sens Wireless Netw 2005;1(1–2):27–39.

Cai Y, Li M, Shu W, Wu MY. “ACOS: an area-based collaborative sleeping protocol for wireless sensor networks”. Ad Hoc Sens Wireless Netw 2007;3:77–97.

Gallais A, Carle J. “An adaptive localized algorithm for multiple sensor area coverage”. Ad Hoc Sens Wireless Netw 2007;4:271–288.

Gallais A, Carle J. “Performance evaluation and enhancement of surface coverage relay protocol”, LNCS 4982. Singapore: IFIP Networking; 2008. pp. 124–134.

Gallais A, Carle J, Simplot-Ryl D, Stojmenovic I. “Ensuring k-coverage in wireless sensor networks under realistic physical layer assumptions”. Proceedings of the 5th IEEE International Conference on Sensors; Daegu, Korea; 2006. pp. 880–883.

Gallais A, Carle J, Simplot-Ryl D, Stojmenovic I. "Localized sensor area coverage with low communication overhead". IEEE Trans Mobile Comput 2008;7(5):1–12.

Gu L, Stankovic JA. "Radio-triggered wake-up capability for sensor networks". Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'04); Toronto, Canada; 2004.

Gui C, Mohapatra P. "Power conservation and quality of surveillance in target tracking sensor networks". ACM Mobicom; Philadelphia, USA; 2004.

Gupta H, Das SR, Gu Q. "Connected sensor cover: self-organization of sensor networks for efficient query execution". Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc); Annapolis, USA; 2003.

Huang C, Tseng Y. "The coverage problem in a wireless sensor network". Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications; San Diego, USA; 2003. pp. 115–121.

Jiang J, Dou W. "A coverage preserving density control algorithm for wireless sensor networks". Proceedings Third International Conference Ad-Hoc Networks and Wireless (ADHOC-NOW); Vancouver, British Columbia; 2004.

Kumar VSA, Arya S, Ramesh H. "Hardness of set cover with intersection". Proceedings of the 27th International Colloquium on Automata, Languages and Programming; Geneva, Switzerland; 2000. pp. 624–635.

Kuruvila J, Nayak A, Stojmenovic I. "Hop count optimal position based packet routing algorithms for Ad hoc wireless networks with a realistic physical layer". Proceedings of IEEE Mobile Ad Hoc and Sensor Systems (MASS); Fort Lauderdale, Florida, USA; 2004.

Li Q, Rus D. "Global clock synchronization in sensor networks". Proceedings of IEEE Infocom 2004; Hong Kong; 2004.

Ovalle-Martinez F, Nayak A, Stojmenovic I, Carle J, Simplot-Ryl D "Area based beaconless reliable broadcasting in Ad hoc and sensor networks". Int J Sens Netw (IJSNet) 2006;2(2): 147–159.

Romer K, Blum P, Meier L. "Time synchronization and calibration in wireless sensor networks". In: Stojmenovic I, editor. Handbook of sensor networks. John Wiley & Sons; 2005. pp. 199–238.

Sheu JP, Tu SC, Yu CH "A distributed query protocol in wireless sensor networks". Wireless Personal Commun 2007;41(4):449–464.

Simplot-Ryl D, Stojmenovic I, Wu J. "Energy efficient backbone construction, broadcasting, and area coverage in sensor networks". In: Stojmenovic I, editor. Handbook of sensor networks: algorithms and architectures. Wiley; 2005. pp. 343–379.

Tian D. "Node activity scheduling scheme in large-scale wireless sensor networks" [PhD thesis]: SITE, University of Ottawa; 2004.

Tian D, Georganas N. "A coverage-preserving node scheduling scheme for large wireless sensor networks". Proceedings 1st ACM International Conference Wireless Sensor Networks and Applications (WSNA'02); Atlanta, USA; 2002. pp. 32–41.

Tian D, Georganas ND "Location and calculation-free node scheduling schemes in large wireless sensor networks". Ad Hoc Netw 2004;2(1):65–85.

Wang X, Xing G, Zhang Y, Lu C, Pless R, Gill CD. "Integrated coverage and connectivity configuration in wireless sensor networks". Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys'03); Los Angeles, USA; 2003.

Wu J, Yang S. "Coverage and connectivity in sensor networks with adjustable ranges". Proceedings of the 2004 International Workshop on Mobile and Wireless Networks (MWN); Montreal, Canada; 2004 Aug.

Ye F, Zhong G, Cheng J, Lu S, Zhang L. “PEAS: a robust energy conserving protocol for long-lived sensor networks”. Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS); Providence, USA; 2003. p. 28.

Younis O, Ramasubramanian S, Krunz M “Operational range assignment in sensor and actor networks”. Ad Hoc Sens Wireless Netw 2008;5(1–2):69–100.

Zhang H, Hou JC. “Maintaining sensing coverage and connectivity in large sensor networks”. Ad Hoc Sens Wireless Netw 2005;1:89–124.

Chapter 4

Geographic Routing in Wireless Sensor and Actuator Networks

Hai Liu[1], **Amiya Nayak**[2], **and Ivan Stojmenovic**[2]

[1]*Hong Kong Baptist University, Hong Kong, P.R. China*
[2]*School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5*

Abstract

Position information enables development of localized routing methods where greedy routing decisions are made at each node, based solely on knowledge of positions of neighbors and the destination, with considerable savings in the communication overhead. Power consumption can be taken into account in the routing process. This chapter will survey existing flooding-based and position-based routing schemes. It also describes a general cost to progress ratio-based approach for designing routing protocols under a variety of metrics, such as hop count, power, remaining energy, delay, and others. The chapter also describes routing with guaranteed delivery for unit disk graphs and an ideal medium access control (MAC) layer. Gabriel graph, as a localized planar and connected structure needed for such solutions is described. Solutions are expanded toward beaconless behavior, where nodes are not aware of their neighborhood. Georouting with virtual coordinates is based on hop distances to some landmarks. This chapter also discusses the physical layer aspects of georouting, routing in sensor-actuator networks, and the load-balancing issue in routing.

4.1 FLOODING-BASED ROUTING AND GEOROUTING IN SENSOR NETWORKS

Sensors report their measurement to a monitoring station, also called a *base station*, or simply, a *sink*. Individual reporting of discovered events are normally done by a routing task, from a sensor to the sink. In a routing task, a packet is to be sent from a source node to a destination node, via some intermediate nodes in a given multihop network. In wireless sensor networks, the source is normally a sensor while the destination is a sink. In sensor-actuator networks, the actuator (actor) may serve as a source and/or destination node. Sinks, actuators, and even sensors could be fixed or mobile.

Existing practical implementations of sensor networks normally avoid the use of position information due to current technological difficulties in providing it to sensors with sufficient accuracy. Routing is then based on flooding as a step to find a route. Flooding was covered in Chapter 2. It will be reviewed again in Chapter 6 to illustrate data gathering and data aggregation by constructing a tree centered at the sink/actuator. Monitoring center floods route discovery (short) message to all sensors located inside a region. Sensors establish links toward the sensor from which the first copy of the packet is received, and use that link for reporting, or forwarding reports received from neighbors. Thus, effectively, sensors report along the reverse broadcast tree. In wireless *ad hoc* networks, this method is currently a candidate for being a standard, with mobile nodes such as laptops, palmtops, and mobile phones serving as sources, destinations, and intermediate nodes. The source then floods the network, and the destination node replies back to the source upon receiving discovery message(s) using memorized hops (AODV) (Perkins *et al*., 1999) or paths (DSR) (Johnson *et al*., 1996). This method was applied to sensor networks, and is often cited as "directed diffusion" following a description under such a name in Intanagonwiwat *et al*. (2000). There are many variants of this basic method, including multipath construction, quality of service (QoS) provision, and so on. The route discovery message may contain accumulated delay, congestion, power (i.e., a prespecified cost metric) along paths. The best path is then selected at the destination node. A number of local route maintenance techniques are proposed to handle the dynamic nature of the network; however, the maintenance is generally expensive, and often is done by triggering another network-wide flooding. To reduce the area being flooded, the expanding ring search is often applied, which floods in a restricted area, assuming that the destination will be found there; otherwise, the searched area size is increased (e.g., doubled).

Ding *et al*. (2003) considered the problem of finding a route from a sensor to the single sink in a wireless sensor network. Following a reactive route discovery strategy, the sink floods the network and sets the routes. The difference is that each sensor does not memorize the whole route, or a single pointer to predecessor sensor on the route, but instead it memorizes its hop count distance to the sink. When a packet is sent toward the sink, any neighbor at one less hop distance can forward it, instead of reporting back to the first node that sent the task assignment packet to it. For instance, the report can be sent to the neighbor with the highest energy and the smaller hop count, or any neighbor that sent the packet with smaller hop count from the sink (Ding *et al.*, 2003). The node can memorize a few such alternatives during setup phase and try them one by one. Alternatively, a neighbor at one less hop distance can simply retransmit, and the node can block further retransmissions by a separate blocking packet.

Because the geographical location of an event is an important information, position information of sensors is considered available, while recognizing the difficulty in gathering it with reasonable accuracy. Geographic routing (georouting) is the strategy that employs geography information of nodes when routing from the source to the destination. It assumes that nodes in the networks are provided with GPS (global positioning system) devices (Hofmann-Wellenhof *et al.*, 1997), or some localization techniques (Bachrach *et al.*, 2005) are available to obtain the location information of nodes. The nodes exchange location information with their neighbors, and forward packets based on the location information of their neighbors and the destination. It allows routers to be nearly stateless, since packet forwarding is based on location information of candidate neighbors and the location of the final destination only. In wireless sensor actuator networks (WSANs), the destination is normally a sink or an actuator. Location of the destination is flooded over the network to all sensors at the initialization stage of the system. As nodes are not required to maintain routing tables and routing decisions are made based on geographic information, the routing information grows with the density of the network, for example, average node degree, rather than size of the network, for example, total number of nodes. Therefore, the geographic routing algorithms are normally characteristic of low computation complexity and high scalability, which are desirable in large-scale wireless networks. By their localized nature, geographic routing algorithms are highly scalable solutions that do not require any additional control overhead when network topology changes due to energy-conserving sleep cycles (Frey *et al.*, 2005). However, highly mobile networks are difficult to handle since geographic routing is normally based on locations of the destination and neighboring nodes. For instance, obtaining an accurate location of a mobile destination is even more difficult than routing itself (Stojmenovic, 2002). This chapter will deal only with the case of static sinks as or actuator destinations.

4.2 GREEDY, PROJECTION, AND DIRECTION-BASED ROUTING

The simplest form of geographic routing is *greedy* routing which was first described by Finn (1987). In the greedy routing algorithm, each node in the route forwards packets to the neighbor that is the closest to the destination among its neighbors. Only the neighbors that are closer to the destination than the current node are considered. The algorithm is illustrated in Figure 4.1. Suppose node S is the current node in the route and node D is the destination. All neighbors

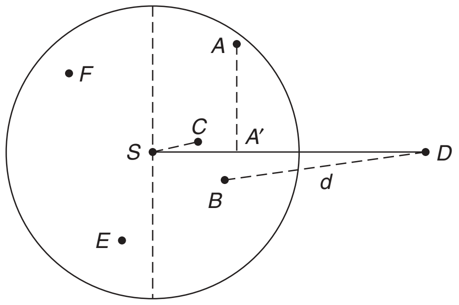


Figure 4.1 Greedy, projection, and direction-based routing.

of S are within the circle centered at S. Suppose node B is the closest to destination D among all S's neighbors and d is the distance between B and D. According to the greedy routing algorithm, node S selects node B as the next hop forwarding node. Greedy routing is suitable for large-scale networks with high density and frequent topology changes because it is simple and localized. The distance to the destination is minimized in each hop of the routing. The *GEDIR* (Stojmenovic and Lin, 2001a) algorithm is a variant of greedy routing. It considers all neighbors (even in backward direction) and selects the node that is the closest to the destination. A message is dropped if it would be sent back to the node where it was previously received from. Another variant is to select the nearest neighbor among those that are closer to the destination than the current node [nearest closer (NC) method, proposed in Stojmenovic and Lin (2001b).

The first geographic routing was described by Takagi and Kleinrock (1984). The notion of *progress* was introduced to define the *most forward within radius* (MFR) greedy routing algorithm. Suppose A is a neighbor of S. The progress of A to D corresponds to the dot product $SA \cdot SD = |SA^{'}||SD|$ of two vectors SD and SA, where $SA^{'}$ is the projection of SA onto line SD, and $|XY|$ is the Euclidean distance between X and Y (dot product formulation of the progress is given in Stojmenovic and Lin (2001a)). The MFR algorithm considers all neighbors of S and selects the node A, such that $SA \cdot SD$ is maximized. That is, the (signed) length of projection $|SA^{'}|$ is maximal. Then, node S forwards packets to node A. Only neighbors with positive progress are considered in MFR. The nearest neighbor with forward progress (NFP) method (Hou and Li, 1986) selects the nearest neighbor among nodes with positive progress.

Another strategy of geographic routing utilizes direction information of next hop candidates with respect to the line toward the destination. Kranakis, Singh, and Urrutia proposed *compass routing* (also referred to as the *DIR* method) in Kranakis *et al.* (1999). It selects the next hop by minimizing the angle $\angle ASD$ between lines toward candidate neighbor A and destination D. In the example in Figure 4.1, $\angle CSD$ is the minimum such angle among all S's neighbors, and node C is selected as the forwarding node.

Stojmenovic and Lin (2001a) proved that greedy, GEDIR, and MFR routing are loop-free while DIR routing is not. Greedy routing selects the neighbor that is closer to the destination than the current node. There is no backtracking and thus it is loop-free. The proof that MFR is loop-free follows. Suppose $A_1, A_2, \ldots, A_n$ are the nodes in the loop such that A_1 forwards packets to A_2, A_2 forwards packets to $A_3, \ldots, A_{n-1}$ forwards packets to A_n and A_n forwards packets to A_1. At the current node S, MFR selects the neighbor A for which $AD{\cdot}SD$ is minimized, that is, $SA{\cdot}SD$ is maximized. We have $DA_n{\cdot}DA_1 < DA_2{\cdot}DA_1 = DA_1{\cdot}DA_2$, since node A_1 forwards packets to A_2 not A_n. Similarly, it follows $DA_1{\cdot}DA_2 < DA_2{\cdot}DA_3 < \ldots < DA_{n-1}{\cdot}DA_n < DA_n{\cdot}DA_1$. This is a contradiction. Therefore, MFR routing is loop-free.

The example in Figure 4.2 shows that direction-based routing cannot guarantee loop-free routes in UDGs. Suppose S is the source node, and B is not its neighbor (transmission radius is shown in the figure). According to direction-based routing, node S selects neighbor A as its deviation from the line SD is smaller than that of C. Similarly, node A selects B, B selects C, and C selects S. Thus, direction-based routing enters the loop $S \rightarrow A \rightarrow B \rightarrow C \rightarrow S$.

Five routing algorithms: greedy, MFR, direction (compass), shortest path(in terms of hop count), and NC routing are illustrated in Figure 4.3. The task is to find a route from the source S to the destination D.

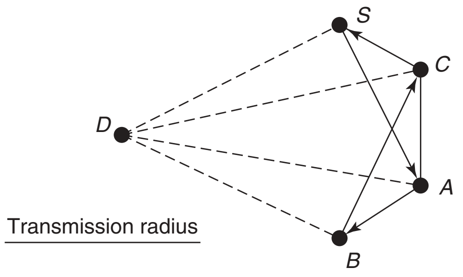


Figure 4.2 A counter example for direction-based routing.

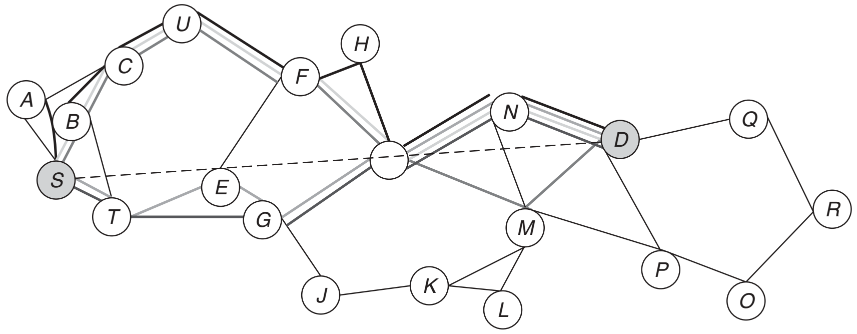


Figure 4.3 Paths with different routing algorithms: greedy: $S \rightarrow C \rightarrow U \rightarrow F \rightarrow I \rightarrow N \rightarrow D$, MFR: $S \rightarrow C \rightarrow U \rightarrow F \rightarrow I \rightarrow M \rightarrow D$, DIR: $S \rightarrow T \rightarrow E \rightarrow G \rightarrow I \rightarrow N \rightarrow D$, shortest path: $S \rightarrow T \rightarrow G \rightarrow I \rightarrow N \rightarrow D$, NC: $S \rightarrow B \rightarrow C \rightarrow U \rightarrow F \rightarrow H \rightarrow I \rightarrow N \rightarrow D$.

4.3 APPLICATIONS OF COST TO PROGRESS RATIO FRAMEWORK TO GEOROUTING

Stojmenovic (2006) proposed a framework for designing network layer protocols for sensor networks including localized routing, broadcasting, area coverage, and so on. The framework is based on optimizing the ratio of the cost to progress, where the cost to reach the next hop forwarding node in routing is expressed in a certain metric, and the progress is a measure of advance toward the destination.

Examples of cost metric are hop count, power, reluctance, power * reluctance, delay, and expected hop count (Stojmenovic, 2006) (see also Chapter 1). Each link has a cost measure, which depends on the assumptions and metrics used. The framework assumes that each node knows the cost of each of its links to the neighboring nodes. The basic idea of the framework is as follows. Suppose the source or current node S has k neighbors, where only neighbors closer to the destination than the current node are considered to ensure progress at each step. That is, S has k choices to forward a packet toward the destination. Node S then computes C_i/P_i, $i = 1, 2, \ldots, k$ for each neighbor, where C_i and P_i are the cost and progress, respectively, of ith candidate neighbor. The neighbor with the minimum cost to progress ratio is selected to forward the packet. The same rule is continuously applied by the receiving node to select the next hop. The routing process continues until the destination is reached or no neighbors with progress are available. If no such neighbor exists, the packet is dropped or a recovery scheme, based on the specific cost metric used, is applied to make a progress before resuming the scheme. The framework was illustrated by applying it to the following well-known geographic routing algorithms.

In the greedy routing (Finn, 1987) introduced in the previous section, the cost metric is the hop count (the number of transmissions on a route) and the progress made by forwarding is reduction of distance to the destination. For the current node that holds a packet, the cost to transmit to any of its neighbors is the same, that is, one hop. In the example in Figure 4.1, the cost to progress ratio for node B is $1/(|SD| - |BD|)$. $|SD| - |BD|$ is to be then maximized, as in the greedy routing algorithm. If the progress metric is defined as the projection of neighbors on line SD, the routing algorithm becomes the MFR (Takagi and Kleinrock, 1984).

Another example is the localized power-aware routing, which was described by Kuruvila *et al.* (2004). In Figure 4.4, the power required for node C to reach node A is proportional to $|\mathrm{CA}|^\alpha + c$, where α is power attenuation factor, which is normally between 2 and 6, and c is a constant. The constant c accounts for the energy and minimal signal strength for correct signal reception. This power measure is used as the cost. The progress is defined as $|CD| - |AD|$ (only positive progress is considered, $|CD| > |AD|$). Thus, the cost to progress ratio of the power-aware routing is $(|\mathrm{CA}|^\alpha + c)/(|\mathrm{CD}| - |\mathrm{AD}|)$. The selected neighbor minimizes the power spent per unit of progress made in terms of getting closer to the destination.

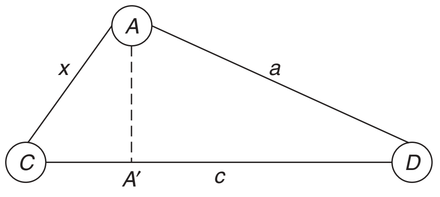


Figure 4.4 Current node C, candidate neighbor A, and destination D.

Note that power-aware routing may result in early energy depletion of certain nodes. If residual energy of nodes is included into cost as reluctance, the goal of routing is to maximize the number of routing tasks the network can perform. For instance, let $f(A)$ denote the inverse $1/g(A)$ of the normalized (in interval [0,1]) residual energy $g(A)$ in node A. Nodes with more residual energy, that is, smaller $f(A)$, are more eager to forward packets while nodes with less residual energy, that is, larger $f(A)$, are reluctant to do so. The routing algorithm selects a neighbor A that minimizes $f(A)/(|CD| - |\mathrm{AD}|)$, subject to $|CD| > |AD|$. This routing framework can be applied to other cost metrics, such as QoS requirements, transmission delay, link quality, and data.

All routing protocols based on the cost to progress ratio can be improved by applying the *iterative improvement* method, which was described by Hang *et al.* (2004) (for QoS metric costs) and (Kuruvila *et al.*, 2004). Suppose current node C selects neighbor A to minimize the cost to progress ratio while the overall goal is to minimize the total sum of costs over the route. If there is another neighbor B of node C, such that $cost(CB) + cost(BA) < cost(CA)$ then it could be more beneficial to forward the packet to node B instead of node A. Such improvement could be iteratively repeated until no improvement is possible. Note that the improvement may be locally done at node C without message exchange if node C has needed information for a given metric (it involves metric between two neighbors).

The iterative improvement is a special case of the general scheme (Sanchez and Ruiz, 2006; Wu and Candan, 2007; Elhafsi *et al.*, 2008) (with power consumption as the metric) based on shortest weighted paths toward temporary destination. Suppose that current node C selects neighbor A having the best cost to progress ratio (in the first proposal (Sanchez and Ruiz, 2006), temporary destination is decided by hop count-based greedy routing). Instead of sending the message directly to A, node C constructs the shortest cost path (using the same cost metric as weight) from C to A, and forwards the packet to the first node B on that path (often $B = A$) to minimize the overall cost. That node then applies the same reasoning, starting from selecting its own temporary destination (Elhafsi *et al.*, 2008), or keeping the same destination until it is reached (Wu and Candan, 2007). To avoid loops between two neighboring nodes, only neighbors directly connected to the temporary destination are considered. If the temporary destination is fixed until it is reached (Wu and Candan, 2007) then progress toward final destination at each step is not verified. Otherwise, to avoid loops, only nodes closer to final destination (not only temporary one) can participate in the shortest path (Elhafsi *et al.*, 2008).

If greedy routing (with a given cost metric) cannot make progress at a given node, recovery mode is invoked. Recovery mode (covered later in this chapter) uses hop count as a metric to guarantee recovery [such resolution is proposed in Stojmenovic and Datta (2004). However, the total cost of following these predetermined edges is then not optimized with respect to the given cost metric. In algorithms (Wu and Candan, 2007; Elhafsi *et al.*, 2008), these edges are indirectly followed, by replacing direct transmission between end points of these edges with shortest weighted paths between them. Further, Elhafsi *et al.*, 2008 builds a connected dominating set (CDS) from a given set of nodes, and computes its Gabriel graph (GG) to obtain the planar graph G'. Face routing is applied on G' only to decide which edges to follow in the recovery process. On each edge, shortest weighted path routing is applied. Then the next edge is similarly followed, until recovery is possible. This two-phase (greedy-face) routing process reiterates until the final destination is reached.

One of the major advantages of the cost to progress ratio framework is that it has no added parameters such as thresholds. In a typical threshold-based approach, "bad" links are eliminated, and the packet is dropped if there is no "good" neighbor. However, a reasonable path may contain only one weak "bridge." Experiments conducted so far indicate that threshold-based approaches are inferior for all threshold values, because of either high failure rate if threshold is too restrictive or suboptimal path choices when generous threshold choice allows one or more very weak links into a path, creating a bottleneck for the route. This occurs because final selection over "acceptable" links is made normally by using a metric different from the cost metric. For example, the node closest to the destination could be chosen, despite its barely acceptable status in terms of the selected metric, such as delay.

'Load-balancing is needed to effectively use available sources and keep the nodes' energy consumption balanced by equally distributing the load. The problem is to route data packets avoiding the congested path so as to balance traffic load over the network and lower end-to-end delay. Distributing the load within the network has two advantages. First, resource of the network is fully utilized through distributing network load. An efficient load-balancing routing protocol is able to improve packet delivery rate and network throughput. Second, energy consumption is balanced by equally distributed load, so that the network lifetime can be prolonged. A dynamic parameterless load-balancing georouting protocol has been proposed in Stojmenovic (in preparation). The node holding the packet for delivery compares costs of sending the packet to all available neighbors that are closer to the destination and not fully loaded, against the progress made. The neighbor with the minimum cost over progress ratio $Load(A)/(|SD| - |AD|)$ is selected. In this formulation, the load (as selected cost) could be the ratio of already consumed bandwidth over total bandwidth at node A. The cost is then increasing linearly with the consumed bandwidth. A more progressive cost can be used by defining $Load(A)= 1/capacity(A)$, where $capacity(A)$ is the normalized remaining bandwidth (capacity) at neighboring node A (Stojmenovic, in preparation).

4.4 MEMORIZATION-BASED GEOROUTING WITH GUARANTEED DELIVERY

Greedy based routing stops if a current node cannot find any neighbor that is making an advance with respect to the selected advance mechanism, such as reducing the distance to the destination. However, a route from source to destination may still exist. We consider only localized methods to find the route when the selected greedy technique fails. They are generally divided into two classes, depending on whether or not any information about the route has been left at visited nodes, for possible later consultation. This is not allowed in memoryless routing, where all needed information is included in the packet. We will first cover some techniques that do allow memorization. In some cases, memorization can be justified. One example is the creation of a path between two nodes that will be used for an ongoing traffic between them (e.g., for QoS-based applications), where nodes on the path need to memorize the next hop. The alternative is, obviously, to record the whole path in the message; but, with increased path lengths this method does not scale well, as increased message size increases collisions and reduces bandwidth. We will describe several recovery mechanisms based on memorization.

Stojmenovic and Lin (2001a) proposed flooding-based methods, called *f-greedy* and *f-MFR*, which apply *greedy* routing and *MFR* at intermediate nodes and run a recovery mechanism at concave nodes. Each concave node memorizes message IDs and rejects further copies of the same message. That is, neighbors of concave node C learn about C's concave status from the packet and do not select C as the next hop forwarding node in future attempts. Each neighbor of the concave node initiates a separate routing task toward the destination, in parallel. Some paths can be terminated later when reaching a node already handling the same routing task for a previous path via that node.

A localized depth first search (DFS)-based routing algorithm was proposed by Stojmenovic *et al.* (2002). Depth first search, which is different from *f-greedy*, is single-path routing. Each node remembers if it has already been visited by the DFS traversal, and the node from where the message was received for the first time. Packet forwarding is performed by sorting all neighbor nodes according to their distance from the destination D and selecting the node that is the closest to D. As already-visited nodes have already transmitted a forward packet, which is overheard by their neighbors, the neighbors can learn their status and do not select them for forwarding again (Vukojevic *et al.*, 2008). A returned message will be sent to the next choice in the sorted list of all next hop nodes. If all neighbors are explored and return the message, the message is returned to the node from which it was first received. The DFS method (Vukojevic *et al.*, 2008) is applied to arbitrary cost metric. Neighbors of the current node are sorted based on the cost to progress ratio that they can provide, and used in that order in attempts to find a route.

In the example in Figure 4.5, according to the DFS algorithm in Stojmenovic *et al.* (2002), source S sorts three neighbors according to the distance to D, and

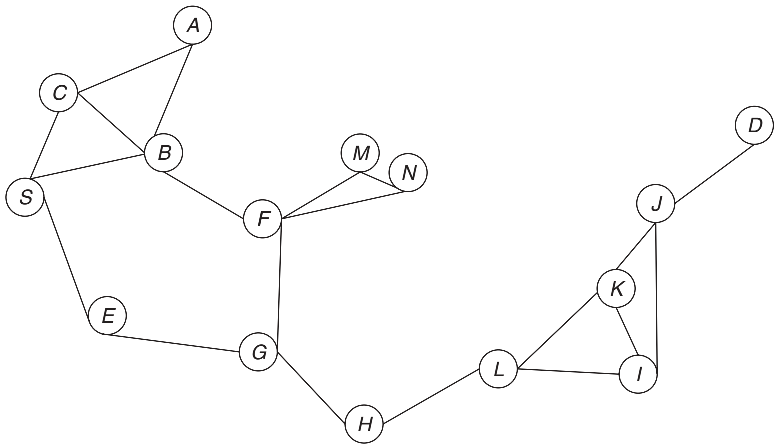


Figure 4.5 DFS routing before (S ***BFNMF*** *MNF* ***GHLKJD***) and after (S ***BFNM*** *NF* ***GHLKJD***) enhancements.

then forwards to B, which is closest to D. Similarly, B sorts its neighbors A, C, and F, excluding sender S, and forwards to F. F forwards to N, which then forwards to M. M forwards to F, which has been visited. Thus, F rejects the message to M. M returns to N since it has no available neighbors. Similarly, N returns to F. F does not forward to its second neighbor M since it received the forwarding message from M already. Instead, F forwards to its last neighbor G. G forwards to H, which ultimately reaches D via $HLKJD$. The routing path of DFS before enhancement is *SBFNMFMNFGHLKJD*. In the enhanced version of DFS (Vukojevic *et al.*, 2008), 'nodes can learn their neighbors' status by overhearing the transmissions of already-visited neighbors. In the example, M eliminates both F and N from its available neighbor list after F forwards to N. F eliminates both N and M from its available neighbor list after N forwards to M. So the routing path of DFS after enhancement is *SBFNMNFGHLKJD*.

The method, however, does not work well in "island" areas with dense node population, as all these nodes would be visited before exiting the island to try another route toward destination. To reduce this problem, DFS is applied only on nodes from a CDS (Vukojevic *et al.*, 2008).

Application of DFS routing for QoS support was further discussed in Stojmenovic *et al.* (2002). On the basis of the information about its own physical location and periodically updated location information of all neighbors, each node is able to estimate the current speed of any neighbor and estimate how long the link will exist. The information could be used to find a route that supports a specified connection-time requirement. In addition, a minimum bandwidth requirement and maximum delay may be considered as well during DFS traversal. A message is returned once the maximum delay is exceeded or no outgoing edge matches the minimum bandwidth requirement. Intermediate nodes along the path memorize the uplink and downlink of the path, such that the QoS communication between the source and the destination can be set up.

Ma *et al.* (2008) proposed a detouring mode for any geographical routing protocol (but it has no impact on greedy routing). The strategy is applied to prune the path found by the georouting protocol when the first packet is routed. After the first packet is delivered, a pruned path is also obtained, and subsequent packets can be forwarded using the pruned path. Suppose that a node A forwards a packet to its neighbor B, and afterwards hears the same packet being forwarded by another neighbor C, node A can immediately make a shortcut by forwarding other packets for same destination to C directly, bypassing at least node B. The algorithm requires some state information to be recorded at nodes that make shortcuts. Similar bypassing algorithm was previously applied for the specific case of DFS routing algorithms in Stojmenovic *et al.* (2002), as natural part of DFS process, when $C = A$, as part of constructing QoS route out of initial route. The bypassing algorithm can be extended to allow common neighbors to intervene in the path. Suppose that node B hears a packet being forwarded by its neighbor A, and later on by its neighbor C, with hop count being increased by more than 2. Then node B may offer node A to forward future packets, which will be then delivered to neighbor C, thus making detour with two hops from A to C.

4.5 GUARANTEED DELIVERY WITHOUT MEMORIZATION

4.5.1 Face Routing in Planar Geometric Graphs

A *planar graph* is a graph that can be drawn on the plane in such a way that edges intersect only at their end points. Figure 4.6 shows a planar graph. In *geometric graphs*, each node is aware of its position and those of its neighbors, and therefore angles toward them. Examples of planar and geometric graphs include street maps, or rooms on the same floor in a building. Planar geometric graphs divide graph into faces. In the example in Figure 4.6, face F_1 is polygon *SABC* and face F_2 is polygon *BEFGHC*. Kranakis *et al.* (1999) described the first localized memoryless routing algorithm for planar geometric graphs, which guarantees delivery whenever the source and destination are connected.

The face routing in Bose *et al.* (1999) is an improvement on the routing algorithm in Kranakis *et al.* (1999). The main idea of the face routing (Bose *et al.*, 1999) is to advance (toward destination D) intersections of faces with a straight line segment that connects the last intersection X (initially it is the source S) and the destination D. A packet is routed along the interiors of the faces until an edge on the route intersects *XD* between X and D. In Figure 4.6, the line intersects the planar graph with faces $F_1, F_2, \ldots, F_7$. The boundary of any face can be traversed by applying the right-hand rule (counterclockwise traversal) or the left-hand rule (clockwise traversal). In the right-hand rule, the face is traversed by keeping the right hand on the wall while walking forward. That is, the packet is forwarded along the next edge counterclockwise from the edge where it arrived. When the packet arrives at an edge intersecting the line

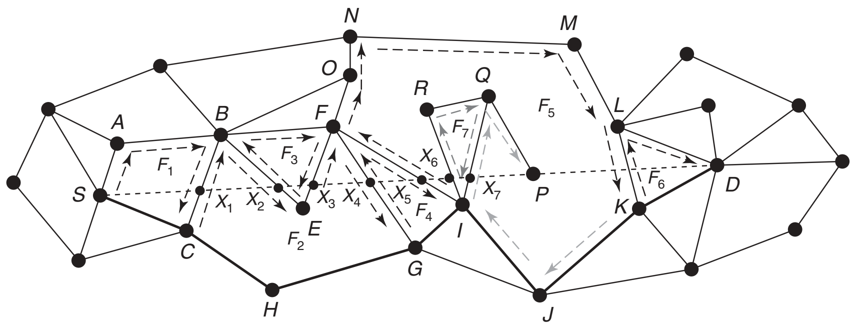


Figure 4.6 Face routing.

XD, the next face intersected by the line is handled in the same way. The process continues until the packet reaches the destination *D* or the first edge of current face traversal is traversed twice in the same direction (this case indicates that source and destination are disconnected, and a loop is created).

Consider the example in Figure 4.6, and assume the left-hand rule is applied. Face F_1 intersects line *SD* and the packet traverses in F_1 over path *SABC* until edge *BC* intersects line *SD* at point X_1. The next face F_2 is traversed on path *CBE* until the next intersection X_2, followed by face F_3 on path *EBFE* and intersection X_3. The path then follows face F_2 again along *EFG*, F_4 along *GFI*, F_5 along *IFONMLK*, and finally face F_6 until delivery to *D*. The whole path is indicated with dashed edges. This variant normally traverses intersecting edges back and forth and is also known as *after crossing variant*, and can be applied similarly with the right-hand rule on each face. If we want to avoid intersecting edges twice, a before crossing variant can be used. It can also start with the left or right-hand rule, or a choice among them may be based on some other criteria, for example using smaller initial angle/direction toward *D*. Face F_1 is then traversed by edge *SC*, which is closer in direction to *SD* than edge *SA*. The next face F_2 is selected and the reference line is updated to X_1D. The packet is forwarded from *C* to *H* and then to *G* until edge *GF* intersects line X_1D at point X_4. Similarly, the face switches to F_4 and the packet is forwarded from *G* to *I* with edge *IF* intersecting line X_4D at point X_5. Finally, the packet will reach *D* via path *SCHGJKD* in bold line.

Note that the traversed face after the intersection may be the same face traversed before the intersection with *XD*. Suppose the destination is point *P* on line *SD* in Figure 4.6, and suppose that the same after crossing variant (dashed line) is followed. While traversing F_5, at point *K*, the path will continue on F_5 along *KJIQ*. At the intersection X_7 of *IQ* and *SP*, face F_5 is selected again and is traversed by algorithm (Bose *et al*., 1999), since it contains the line X_7P, and the message is delivered to *P* along *IQP* in the after crossing or left-hand variants, or along $X_7JKLMNOFX_7RQP$ line (not shown in the image) in the before crossing or right-hand variants. The algorithm (Karp and Kung, 2000) however forces the change of face at every intersection X_iD (note that the X_i must be internal point on the line segment X_{i-1} D). Then the algorithm (Karp and Kung, 2000) selects face F_7, and the message traverses along *QIR* or *IRQ* indefinitely in a loop, since X_6 is not inside the line segment X_7P. The face change will never occur, resulting in a loop. Thus, this step is correctly implemented in Bose *et al.* (1999), while it has been mistakenly described in Karp and Kung (2000), leading to lack of guaranteed delivery in arbitrary planar geometric graphs.

There are several variants of face routing, primarily addressing various decisions on the choice of left or right-hand rules to use for each traversed face. The differences can also be made based on changing face before crossing an edge on the route, or after crossing it. Few variants of the face algorithm can be described as follows:

Face routing

```
// $S$: source, $D$: destination
X ← S
repeat
    let $f$ be the face of $G$ with $X$ on its boundary that intersects line $XD$
    traverse $f$ (counter)clockwise until reaching an edge $UV$
        that intersects line segment $(X, D)$ at some internal point $Q \neq X$
    $X \leftarrow Q$
    continue routing from node $U$ (before crossing) or $V$ (after crossing $XD$)
until $X = D$.
```

4.5.2 Gabriel Graph

We are interested in face routing in unit disk graphs (UDGs), not in planar geometric graphs. This is because the UDG is utilized to model wireless *ad hoc* and sensor networks. In general, an arbitrary UDG is not planar. We therefore need a geometric structure that will be derived from UDG and will provide a planar graph in localized manner. Currently, the most convenient structure for routing applications is GG.

The GG was first introduced by Gabriel and Sokal (1969). Two points u and v are joined by an edge in the GG whenever the disk with diameter $|uv|$ contains no other points from the given point set. In Figure 4.7a, the dashed circle is the *forbidden region* of GG where there is no other node. Figure 4.8 shows a GG constructed from a UDG. Bold edges belong to GG and all edges belong to UDG. In the 3D definition of GG, two points are joined if the sphere, drawn with them as end points of its diameter, does not contain other points from the set.

The GG concept is applied on top of an UDG. Let S be a set of points in the plane and $U(S)$ the UDG, which contains all points in S. Let $GG(S)$ denote the GG induced by S. The following theorem shows that the GG preserves the connectivity of UDG.

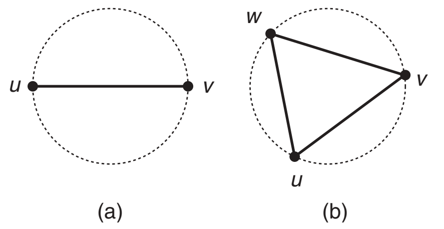


Figure 4.7 Forbidden region for (a) Gabriel graph: (b) Delaunay triangulation.

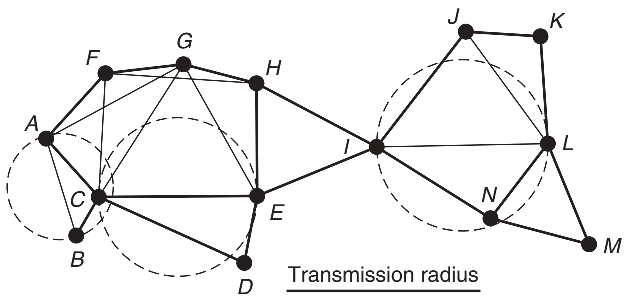


Figure 4.8 Gabriel graph (bold edges) and unit disk graph (all edges).

Theorem 4.1. **(Bose *et al.*, 1999)** If *U(S)* is connected, then *GG(S)* ∩ *U(S)* is connected.

Proof. We will prove that both graphs contain the minimal spanning tree *MST(S)* of the same set S; therefore, their intersection also contains *MST(S)*, and is therefore a connected graph. We will prove *MST(S)* ⊆ *GG(S)* by contradiction. If *MST(S)* is not subset of GG then there exists edge $PQ \in$ *MST(S)*, $PQ \notin$ GG. Since PQ is not in GG, there exists node W inside the disk with diameter PQ. This node W then satisfies $PW < PQ, QW < PQ$. Because PQ is in *MST(S)*, either PW or QW is not in MST. Assume $PW \notin$ *MST(S)*. Replace PQ by PW in *MST(S)*. The new *MST(S)* has smaller sum of edge lengths, which is a contradiction. Therefore *GG(S)* contains *MST(S)*, and is therefore connected. Suppose that the radius of *U(S)* is R. All edges of *U(S)* are thus not greater than R. According to Kruskal's algorithm (Kruskal, 1956), these edges are first considered in constructing *MST(S)* before other edges whose length is greater than R. After considering of all edges in connected *U(S)*, *MST(S)* includes all nodes from S and is already completed. That is, *MST(S)* ⊆ *U(S)*. So, *GG(S)* ∩ *U(S)* is connected. ♦

Theorem 4.2. Gabriel graph is a planar graph.

Proof. Proof is by contradiction. Assume that it is not a planar graph. Then there exist two intersecting edges UV and PQ. From UV, $PQ \in$ *GG(S)*, it follows that $\angle PUQ < \pi/2, \angle PVQ < \pi/2, \angle UPV < \pi/2, \angle UQV < \pi/2$. Then the sum of angles of the quadrilateral $UPVQ$ is $< 2\pi$, which is a contradiction. ♦

The construction of GG is straightforward from its definition. To test whether or not an edge uv belongs to GG, node u can check if distance from other points to the center of line segment UV is $> |UV|/2$. Alternatively, u can verify if angles over uv from each of neighboring points is acute. If so, the edge is in GG, otherwise it is not included in GG. The computational time complexity of this algorithm for testing all edges at a node is $O(d^2)$, where d is the maximum degree in the network. However, the communication in wireless networks is much more expensive than computation. If node u is already aware of the geographic locations of itself and all its neighbors, no additional messages are involved in the construction of GG. This makes GG a localized scheme. Moreover, it is a zero-message planar graph construction method, a very desirable property.

The relative neighborhood graph (RNG) is described in Chapter 2. It has been pointed out (Li *et al.*, 2001) that the number of edges in RNG and GG are bounded by $3n - 10$ and $3n - 8$, respectively, where n is the number of nodes in the networks. Thus, the average node degree in RNG and GG is bounded by 6. In fact, the average degree of RNG is about 2.5 (Hou *et al.*, 2005) while the average degree of GG is about 3.8 (Huang *et al.*, 2004). The maximum node degree of each node of RNG can be reduced to 5 if the length of each edge is made unique. Suppose a node u in RNG has degree greater than 5. There must exist two neighbors v and w, such that $\angle vuw \leq 60^\circ$. Since $|uv| \neq |uw|$ (suppose $|uv| > |uw|$), node w must be inside the forbidden area of uv, contradicting that uv is in RNG. To make all edges unique, it suffices to consider them as records with primary, secondary, and ternary keys being $(|uv|, min(u,v), max(u,v))$.

We will also briefly discuss Delaunay triangulations (DT) as possible planar graph alternatives to GG. An edge uv is in DT if there exists a circle, whose chord is uv, which does not contain any other node in its interior. Alternatively, DT contains all triangles uvw, which satisfy the condition that the circle passing through u, v, and w does not contain any other node (Figure 4.7b). However, DT cannot be constructed by localized algorithms. The reason is that neighbor information of a node alone is not sufficient to determine if any triangle in which the node is an end point belongs to DT. Node u may not be aware of the existence of node x if x is inside the circle but outside the u's communication area. That is, circle sizes in the definitions are not limited, and localized knowledge, then, is insufficient.

From the first definition of DT, we can see that GG $\subseteq$ DT. Suppose that an edge uv belongs to RNG. This means that there is no node w such that $|uv| > |uw|$ and $|uv| > |vw|$. Thus w is not inside the circle with diameter uv. That is, RNG $\subseteq$ GG. Therefore, we have MST $\subseteq$ RNG $\subseteq$ GG $\subseteq$ DT (Hou *et al.*, 2005).

4.5.3 Routing with Gabriel Graph

The GG can serve as a planar graph needed for face routing to work. The main problem with respect to the performance of face routing is exploring significant portions of boundaries of faces. Therefore, Bose *et al.* (1999) proposed a combination of the face routing algorithm with the distance-based greedy routing. The algorithm, which is referred to as *GFG* (greedy-face-greedy), applies the greedy algorithm until the packet reaches a node such that all its neighbors are further from the destination than the node itself. The face routing is applied until the packet reaches another node that is strictly closer to the destination. The greedy algorithm is then resumed. The algorithm can switch between greedy and face mode several times, but guarantees progress and delivery because face routing is always successful, and loops cannot be created since the algorithm always advances in greedy mode, and is guaranteed to further advance while in face mode (that is, it is guaranteed to recover).

The example in Figure 4.9 illustrates how the face routing can route over the void area. Suppose the greedy algorithm is applied and the packet reaches node S, which has no neighbors closer to destination D than itself. Face routing is required for recovery, that is, to find a node that is strictly closer to D than S. If the right-hand rule is applied, the packet follows the shorter route until it reaches the first node B, which is closer to D than node S. Similarly, the packet follows the longer route and reaches node C if the left-hand rule is applied. The greedy algorithm is then resumed after recovery.

The GFG algorithm is illustrated in Figure 4.10, from source S to destination D. Bold edges belong to GG and all edges belong to UDG. Since S cannot find any neighbor which is closer to D than itself, face routing is applied first. Suppose the left-hand rule is used, the packet follows route *SYXTPOK* until it reaches the first node K with $|KD| < |SD|$. The greedy algorithm is then applied and the packet advances following the route *KGHI* until it reaches node I, which has no neighbors closer to D than I. Face routing is applied again and the packet follows route *IHGFECB* until it reaches node B with $|BD| < |ID|$. Note that face routing considers only edges in GG and edge BE is not selected. The greedy algorithm is then applied and the packet finally reaches D via the route *BAD*. If the right-hand rule is applied, the packet follows the route $S \to Z \to S \to Y \to X \to W \to V \to W \to U \to T \to R \to T \to P \to Q \to P \to O \to M \to N \to M \to L \to J \to F$ and reaches the first node F with $|FD| < |SD|$. The greedy algorithm is then applied and the packet follows the route *FEBAD* (edge BE is selected in the greedy routing) until it reaches the destination D.

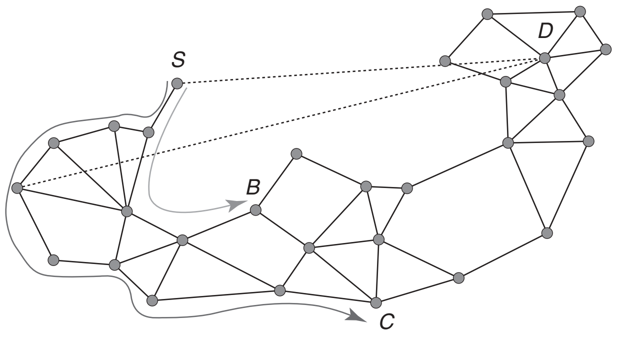


Figure 4.9 Traversing the face until recovery.

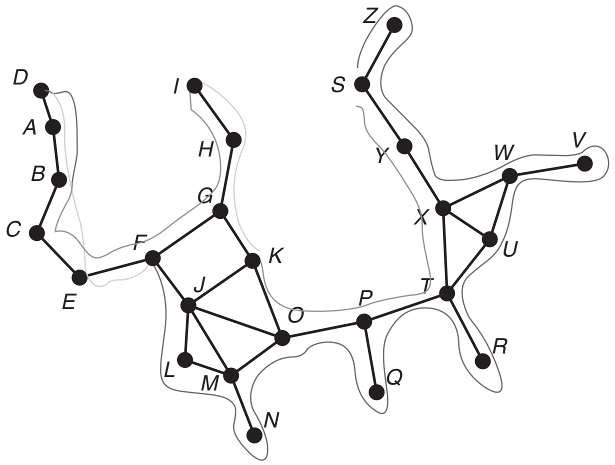


Figure 4.10 Greedy-face-greedy routing from S to D by right-hand and left-hand recoveries.

If the source is disconnected from the destination, GFG will create a loop. A loop is detected if the first edge in face routing is repeated twice by the traversal, in the same direction. The first edge can be added to the message, so that its repetition can be detected. Note that repeating any node on the path is not sufficient to declare a loop.

The variant of GFG using GG as planar graph is expected to have smaller average hop count than the variant with RNG in face routing because GG is denser than RNG (RNG is a subset of GG), and therefore has more edges. More edges lead to fewer edges on faces and therefore to fewer hops.

The GFG algorithm was further improved by Datta *et al.* (2002) to reduce its average hop count. Each forwarding node uses the local two-hop information available to calculate as many hops as possible and forwards the message to the last-known hop directly instead of forwarding it to the next hop. The CDSs technique was further applied and face routing was performed on the dominating set except possibly the first and the last hops.

The GFG algorithm with added IEEE 802.11 medium access layer was later implemented as the greedy perimeter stateless routing (GPSR) protocol by Karp and Kung (2000). The GPSR protocol is a variation of GFG. More precisely, GPSR uses the before crossing instead of the after crossing variant, and discusses the RNG as an alternative to the GG. However, these modifications do not improve the performance of the routing protocol. It was pointed out by Kim *et al.* (2005) that GPSR cannot guarantee delivery in arbitrary planar graphs. The fact was confirmed by Frey and Stojmenovic (2006) and formal proof of delivery guarantee of GFG in arbitrary planar graphs was further given. This is due to a difference in the face routing procedure between GFG and GPSR. The GPSR always switches to the other face whenever the line XD in the algorithm is intersected. However, GFG correctly selected the proper face, which sometimes can be the very same face before the intersection, as illustrated in Figure 4.6, once an intersection point is found. Interestingly, face switch does not occur at all when GG is used as planar graph, which is the reason why the error in the GPSR implementation was not identified before it was reported in Frey and Stojmenovic (2006). More detailed difference between GFG and GPSR could be found in Frey and Stojmenovic (2006). The following theorem shows that under GG, when recovering from a greedy routing failure, it is always possible to reach a node that is closer to the destination than the current node. So, the greedy algorithm can be resumed after traversing only one face with face routing.

Theorem 4.3. (Frey and Stojmenovic, 2006) Let SD be the line between source node S and destination node D in a Gabriel graph G. For any edge UV in GG intersecting the line SD, the distance between D and at least one of the edge end points U or V is smaller than the distance between S and D.

Proof. Since edge UV is in GG, the circle having $|UV|$ as its diameter does not contain nodes S and D. Thus, both angles $\angle USV$ and $\angle UTV$ are less than 90° (Fig. 4.11). Since the angles of the quadrilateral $SUDV$ sum up to 360°, at least one of the angles $\angle SUD$ and $\angle SVD$ is greater than 90°. This makes SD the longest edges in corresponding triangle. Therefore, at least one of the two nodes U or V is located closer to D than the node S. ♦

Therefore, the GFG that uses GG can be implemented by the following simplified algorithm.

```
GFG over GG (Frey and Stojmenovic, 2006)
repeat
    follow greedy until delivery or failure at node $S$
    if failure at $S$ then
        select face $f$ containing the line $SD$
        traverse $f$ until return to greedy is possible
    endif
until delivery
```

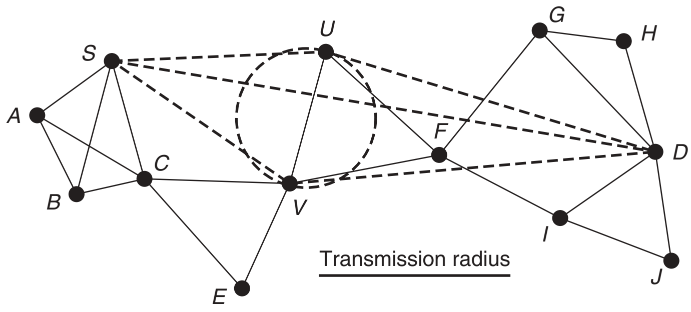


Figure 4.11 Face routing in GG always finds a node closer to the destination (all nondashed edges belong to GG).

Kuhn *et al.* (2003a) proposed an extension of the GFG algorithm which is referred as *greedy other adaptive face routing plus* (GOAFR+). It observes that efficiency of face routing depends on the traversal direction of a face. An improper traversal direction of a face may result in a long path to the destination. So, the basic idea of GOAFR+ is to introduce a circle C centered at the destination and its initial radius is set to include the source. The greedy mode is employed as long as there is a next hop node closer to the destination, and whenever possible the radius of C is exponentially decreased as long as the current visited node is within C. Once the greedy mode reaches a concave node U, a modified version of face routing is invoked. If the face is traversed completely without hitting the circle C, the packet is sent to the node visited so far, which is closer to the destination than U (if no node is closer to the destination than U, routing failure will be reported.). If C is hit for the first time, the face is traversed in the opposite direction. If C is hit for the second time and none of the visited nodes are closer to the destination than U, face traversal continues as if started at U and the radius of C is exponentially increased. Once face traversal visits up to a predetermined constant factor with more nodes closer to the destination, GOAFR+ interrupts face traversal and switches to greedy mode again.

The cost of a path is defined as the sum of costs of all edges in the path, where the cost of an edge could be any cost metric, which is polynomial in the Euclidean distance. In Kuhn *et al.* (2002) it has been shown that the worst case cost of any geometric routing algorithm is bounded by, at most, the quadratic path costs (compared to the shortest weighted path), which is denoted as *asymptotic optimality*. It was pointed out by Kuhn *et al.*, (2003b) that asymptotic optimality cannot be achieved if face traversal is switching back to the greedy mode when the line that connects the concave node and the destination is intersected for the first time (e.g., GFG is not asymptotically optimal). The combination of greedy and face routing becomes asymptotically optimal when packets explore the complete face and switch back to greedy mode at the face edge that is closest to the destination. It has been proven by Kuhn *et al.* (2003b) that GOAFR+ is asymptotic optimal although it does not traverse the complete face in general.

There are significant challenges left in georouting with guaranteed delivery, and hundreds of papers in literature address them. Imprecise location information is a significant challenge for georouting with guaranteed delivery. There is no localized memoryless algorithm for georouting in 3D that has a guaranteed delivery property. Unit disk graphs with equal transmission radii and absence of obstacles are required for GGs to remain connected. An extension for fuzzy UDGs is given in Barriere *et al.* (2001). Two nodes are connected if their distance is smaller than r and are disconnected if the distance is greater than R. Two nodes are randomly connected or disconnected at the distance between r and R. The main algorithm works if $R/r < 1.41$, with certain extensions and message overheads to maintain GG. For other cost metrics, there is still no real alternative to GG-based face routing for recovery mode, which prefers close neighbors. Existing improvements are based on shortcuts, dominating sets, and shortest weighted paths over face traversed edges.

One of the challenging problems is also addressing mobility issues for intermediate nodes on routes, while routing is in progress. The algorithm is loop-free for static networks, but loops can be created by mobile nodes. After entering a face, two nodes on the same face can move close to each other and divide the face into two new faces, leaving a message in one of the faces that does not intersect the imaginary line from source to destination, thus looping forever. However, this problem can be partially resolved (if mobility is not so high that required information becomes unreachable) by adding the time-stamp of the last intersection with the imaginary line *SD* and ignoring links created afterwards.

4.6 BEACONLESS GEOROUTING

The greedy forwarding algorithms normally need to periodically exchange "hello" messages (beaconing) with maximum signal strength by each node in order to broadcast current position information to all one-hop neighbors. The beaconing process of greedy routing costs additional energy consumption, which occurs independently of current data traffic.

Heissenbuttel and Braun proposed the *beaconless routing* (BLR) algorithm in Heissenbuttel and Braun (2004). Beaconless routing was further integrated with the IEEE 802.11 medium access control (MAC) layer in the *contention-based forwarding* (CBF) by Füßler *et al.* (2003) and *implicit geographic forwarding* (IGF) by Blum *et al.* (2003). In BLR, node S currently holding the packet destined for node D will include its own and location of D in the packet, and retransmit either only the request for forwarding or the full message content. Upon receiving the packet, the neighboring node, a candidate for forwarding with a progress, calculates a waiting time-out depending on the relative location coordinates of itself, S, and D. The node located at the "best" location introduces the shortest delays and forwards the packet first (or responds first with the offer to retransmit). The (most) remaining nodes then cancel the scheduled transmission of the same packet.

Beaconless routing is illustrated in Figure 4.12. To ensure that all potential forwarding nodes detect transmission of S, selection of candidate nodes for the next forwarding step is limited in a certain forwarding area. The forwarding area has the property that each node is able to overhear the transmission of any other node in the area. Heissenbuttel and Braun (2004) showed that the circle with a diameter equal to the transmission radius, centered at the line *SD* with S as one end point (the dotted circle in Fig. 4.12) is a good forwarding area with regard to progress and successful hops before greedy routing fails. Several delay functions are investigated, resulting in different forwarding behavior.

A technique called the *active selection method* was further proposed in Füßler *et al.* (2003). A forwarding node sends a control packet instead of the full message to all its neighbors. Neighbors respond with information of their forward progress after a time-out which depends on their distance to the destination. The forwarding node then sends the full message that indicates which of its neighbors will forward the message. Zorzi (2004) proposed to avoid duplicate forwarding in

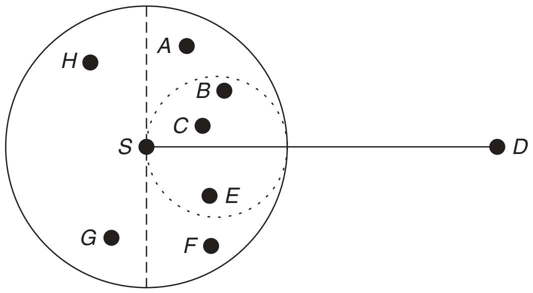


Figure 4.12 Forwarding area in BLR.

a BLR scheme by employing the request-to-send/clear-to-send (RTS/CTS) MAC scheme from IEEE 802.11. The current node sends an RTS signal instead of the message and waits for a CTS signal. If several responses are received, the node selects the one that appears to be the best for forwarding and then sends the message to that neighbor directly.

When greedy routing fails, a recovery strategy such as face routing is required to guarantee delivery. Note that face routing is normally based on a planar subgraph that is constructed from neighborhood information, which is not available in BLR. To solve the problem, Kalosha *et al.* (2008) proposed beaconless georouting schemes with guaranteed delivery. They proposed two solutions: *beaconless forwarder planarization* (BFP) and *angular relaying*. Beaconless forwarder planarization finds correct edges of a local planar subgraph at the forwarder node without hearing from all neighbors. The face routing is then applied in the subgraph. Angular relaying directly determines the next hop of a face traversal. Details of both the schemes are presented next.

Beaconless forwarder planarization is a general scheme that is used to construct GG and RNG. It consists of two phases: *selection phase* and *protest phase*. In the selection phase, the forwarder v broadcasts an RTS including its own location and sets its timer to $t_{\max}$. Each neighbor u sets its contention timer by using following delay function: $t(d) = t_{\max} \times d/r$, where $d = |uv|$, r is the transmission radius and $t_{\max}$ is the maximum time-out. That is, the closer neighbors set smaller waiting times. A node responds with a CTS when its contention timer expires. If a node w receives the CTS of another node w' that lies in the forbidden region $N(v, w)$, w cancels its timer. In the example in Figure 4.13, nodes set their timers in increasing order $w_1, w_2, \ldots, w_6$ according to their distance to node v. Node w_1 responds first. When contention timer of w_2 expires, w_2 responds with a CTS including its location to v. Since w_5 overhears the CTS, it finds w_2 is in the forbidden region $N(v, w_5)$. Thus, w_5 cancels its timer and is referred to as a *hidden node*. Since w_6 does not hear the CTS from both w_4 and w_5, it will respond to a CTS once its contention timer expires. However, edge vw_6 is the violating edge and should not be included in the GG since w_4 and w_5 are in the forbidden region $N(v, w_6)$.

In the protest phase, hidden nodes protest against violating edges. If the set of violating nodes is not empty, the hidden node starts its timer by using the

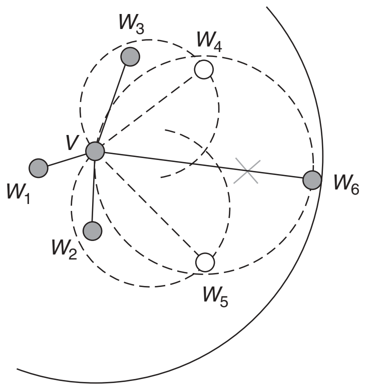


Figure 4.13 Selecting GG edges in BFP.

same delay function. The violating node could be reported by some other hidden node. Otherwise, the hidden node sends the protest message. Upon receiving protests from hidden nodes, the forwarder removes violating edges and finally obtains a planar graph. In the example in Figure 4.13, timer of hidden node w_4 is smaller than the timer of w_5. When timer of w_4 expires, it sends a protest to the forwarder v. v removes the violating edge vw_6. Since w_5 overhears the protest of w_4, it removes w_6 from the set of violating nodes, which then becomes empty. Thus, w_5 remains silent after its timer expires.

Similar to BFP, the angular relaying algorithm consists of selection and protest phases. In the selection phase, the forwarder node v that receives a packet from the previous hop u, sends an RTS including the location of u and itself, and sets its timer to $t_{\max}$. Each neighbor, say w, sets its contention timer by using the following delay function: $t(\theta) = t_{\max} \times \theta/(2\pi)$, where $\theta = \angle uvw$ is in counterclockwise order. A neighbor responds by an "invalid CTS" if it finds other nodes in the forbidden region. The purpose is to let other nodes be aware of its existence. Otherwise they would be hidden and need a chance to protest later. Once the first candidate w answers with a valid CTS, the forwarder immediately sends a SELECT message announcing that w is the first selected node. All candidates with pending CTS answers cancel their timers. The protest phase starts once the first candidate is selected. The forwarder sets its protest timer that covers the time when protests can occur (e.g., $t(\pi/2)$ for GG). No further CTS is allowed. Each candidate node x sets a new timer $t(\theta)$, which determines the order of protests where $\theta = \angle uvx - \angle uvw$. Only nodes in $N(v, w)$ are allowed to protest. Node x that protests automatically becomes the next hop. Afterwards, only nodes in $N(v, x)$ are allowed to protest. Finally, the forwarder sends the data packet to the currently selected candidate after its timer expires.

In the example in Figure 4.14, nodes set their timer in increasing order, $w_1, w_2, \ldots, w_6$. Since w_1 is in region B (the previous hop u is in $N(v, w_1)$), w_1 sends an invalid CTS. Similarly, w_2 sends an invalid CTS since w_1 is in $N(v, w_2)$. After w_3 sends a valid CTS, it is the selected node. However, w_4 protests, and believes it is the next hop. The protest then is sent by w_5. Finally, w_5 is the next hop and v sends the data packet to w_5 directly.

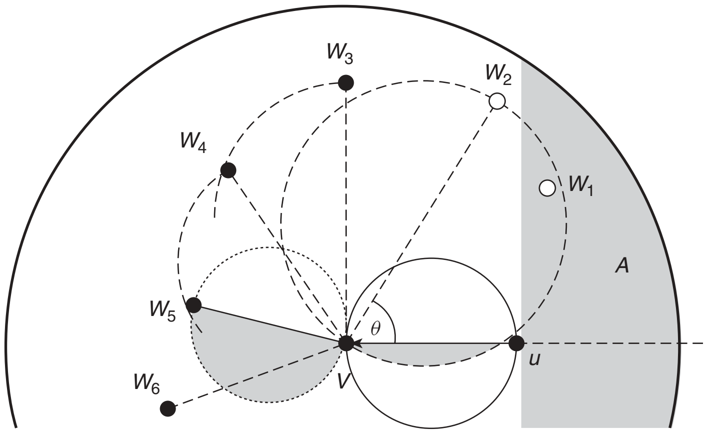


Figure 4.14 Selecting the next hop w_5 after invalid CTS from w_1, w_2, w_3, w_4 in angular relaying.

4.7 GEOROUTING WITH VIRTUAL AND TREE COORDINATES

The accuracy of exact geographic coordinates that is currently available is not sufficient to support the claimed performance of georouting algorithms. An alternative solution is to use virtual coordinates instead of real ones. The typical approach is to assign hop count distances to a certain set of landmark nodes as coordinates of sensor nodes, and define the distance between nodes as the sum of (absolute values of) differences in hop count toward these landmarks (this is also called *Hamming distance*). Greedy routing can be applied to these coordinates (Caruso *et al.*, 2005). The problem, however, is that different nodes can have the same coordinates. The sensors are not properly sorted in such coordinates, and greedy routing may lead to a node that is local minima by Hamming distance. A resolution is proposed in Chavez *et al.* (2007) by adding tree coordinates to nodes, rooted from a landmark node. Greedy routing then proceeds by considering only neighbors that provide progress in tree coordinates. The combined set of two virtual coordinates then enables routing with guaranteed delivery.

Mitton *et al.* (2008) consider the problem of designing power efficient routing with guaranteed delivery for sensor networks with known distances between neighbors but unknown geographic locations. They proposed HECTOR, a hybrid energy-efficient tree-based optimized routing protocol, based on two sets of virtual coordinates. One set is based on rooted tree coordinates, and the other is based on hop distances toward several landmarks. In the algorithm, the node currently holding the packet will forward it to its neighbor that optimizes the ratio of power cost over distance progress with landmark coordinates, among nodes that reduce landmark coordinates and do not increase tree coordinates. If such a node does not exist then forwarding is made to the neighbor that reduces tree-based distance and optimizes power cost over tree distance progress ratio.

4.8 GEOROUTING IN SENSOR AND ACTUATOR NETWORKS

Geographic routing by an adaptive targeting (RAT) protocol was studied by Shah *et al.* (2007). RAT provides sensor-to-actuator communication and dynamic coordination of actuators in response to emergencies. It focuses on applications in which time critical data is required to be sent from sensor nodes to actuators. A sensor node is said to be covered by an actuator if the distance between them is not greater than R. Sensors report only to a single actuator. Actuators broadcast subscribe messages to sensors in their covered areas. When they move, such a message is sent with a frequency based on speed and field dimensions. Actuator-actuator coordination is by broadcasting, whose details are not given.

Routing by adaptive targeting consists of two components: delay-constrained geographic-based routing (DC-GEO) and integrated pull/push (IPP) coordination. In IPP, actuator nodes subscribe to specific events of their interest in the field, and sensor nodes disseminate the event readings to subscribed actuators for time periods of subscription life. Sensor nodes push the data as long as there is any actuator interested for the observed event. Delay-constrained geographic-based routing employs greedy forwarding such that the delay constraint can be met as well as energy consumption of forwarding nodes is balanced.

The process of forwarding the data packets toward actuators consists of two steps. The source node sets the time to live (TTL) field in the packet and each forwarding node updates the TTL by deducting the traversed hop delay. Each sensor constructs a delay-constrained forwarding subset (DCFS) from the neighbors that are closer to the destination, such that the given delay constraint can be met. The second step of forwarding is to balance the load of nodes in DCFS by selecting the forwarding node that has the highest energy level. The proposed routing protocol minimizes energy while meeting delay constraint. The current node, say i, first constructs DCFS from its neighbors for each actuator a, such that the given delay constraint can be met. It applies the criterion $\mathrm{TTL}/T(j) > D(i, a)/D(i, j)$ on each of its neighbor j, where TTL represents the constraint of time remaining to reach the destination, $T(j)$ is the expected delay to relay a packet from i to j, $D(i, a)$ and $D(i, j)$ are distances from current node i to actuator a, and candidate neighbor j, respectively. From DCFS, i selects the next node that has the highest residual energy. Actuators need to be positioned close to emergency area to reduce response time.

When no acceptable neighbor exists (there might be greedy neighbors providing advance but not meeting delay criterion), routing continues in face mode, by variant of GFG (Bose *et al.*, 1999) that follows both faces. The concave node relays the packet to one node on the left and one node on the right along the perimeter of the current face. Then each of these receivers continues with face routing until the next recovery, where the algorithm can return to greedy mode based on delay bounds. Each of future concave nodes on any branch may further branch in two. Authors claimed that one of the branches may loop and denote the failure of that branch. However, this does not follow from the GFG algorithm, which requests a link to be repeated in the same direction, not that merely a node repeats. The algorithm then overall does not guarantee delivery when one exists, because such a node may be on the way to recovery. When GFG works, response time might be improved, but could be also longer because face mode normally involves short edges, and increased number of hops increases delay. Further, one slow link in greedy mode may still lead to a route with an overall acceptable delay. Thus, the algorithm can be revisited.

4.9 LINK QUALITY METRIC IN SENSOR AND ACTUATOR NETWORKS

A resource-aware and link quality-based (RLQ) routing metric for sensor and actuator networks was studied by Gungor *et al.* (2007). Resource-aware and link quality-based routing metric is a combined link cost metric, which is based on both energy efficiency and link quality statistics. Energy efficiency is measured by residual energy and normalized energy cost to transmit and receive a packet. Since actuator nodes normally have higher energy resources than sensor nodes, the energy cost of sensor nodes is assigned with a larger weight. Link quality is measured by the expected number of transmissions, which is calculated from the packet reception rate. The neighbor with the minimum link cost is selected in packet forwarding. The RLQ (Gungor *et al.*, 2007) introduced in the previous section was implemented in the Tmote Sky nodes (http://www.moteiv.com). Two radio hardware link quality metrics are used to measure the link quality during the operation of the network. They are link quality indicator (LQI) and received signal strength indicator (RSSI). Received signal strength indicator is the estimate of the signal power and is calculated over 8 symbol periods, while LQI can be viewed as chip error rate and is calculated over 8 bits following the start frame delimiter. Packet reception rate represents the ratio of the number of successful packets to the total number of packets transmitted over a certain number of transmissions. The test-bed experiments show a strong correlation between the average LQI measurements and packet reception rates. It also shows a good performance of RLQ in terms of packet reception rate, network throughput, and network lifetime.

Souryal and Moayeri (2005) discuss forwarding that adapts to the time-varying channel and exploits spatial diversity to mitigate multipath fading. The routing layer uses long-term measurements of link quality (Signal-to-Noise-Ratio, SNR) to opportunistically select next hop relays on a hop by hop basis. They find the formula for packet success probability as a function of average SNR in quasi-static Raleigh fading. This is multiplied by advance toward destination for selecting the best neighbor. The routing layer will pass M relay candidates to the MAC layer. If $M > 1$, the MAC layer pools the M candidate relays for current Signal-to-Interference-plus-Noise-Ratio (SINR) and position measurements and forwards the packet to the relay that maximizes the expected progress for small-scale adaptivity. To avoid collisions, the relays reply in the order specified by the polling message.

4.10 PHYSICAL LAYER ASPECTS AND CASE STUDIES OF GEOROUTING

Almost all existing literature on geographic routing employs UDG in the communication model. In UDG, two nodes can communicate with each other if and only if their distance is not greater than the common transmission radius. However, as discussed in previous chapters, the UDG model is not realistic since variations of received signal strengths are not considered. It has been pointed out that impact of signal strength fluctuations sometimes is more significant than the impact of node mobility (Stojmenovic *et al.*, 2005). Therefore, reception of a packet is probabilistic. In addition to distance, the received signal strength also depends on other factors, such as environment landscape and transmission medium.

Zorzi and Armaroli (2003) considered advancement as a metric to be used in routing decisions. The advancement provided by a relay node is defined as the difference between the distance of the transmitting node to the intended destination, minus the distance between the relay node and the destination, multiplied by the probability of a successful transmission from the transmitting node to the relay. This idea has been later rediscovered [without citing (Zorzi and Armaroli, 2003)] in several articles, including Kuruvila *et al.* (2005) (conference version from October 2004), Zuniga Zamalloa *et al.* (2008) (conference version in November 2004), and Lee *et al.* (2005).

Kuruvila *et al.* (2005, 2006) proposed geographic routing protocols that are amenable to any realistic physical layer model with fixed and variable packet lengths. Both cases with and without acknowledgments were considered. To employ position-based routing, the first step is to find a reasonably accurate approximation for the bit and packet reception probabilities for the given physical layer model. The lognormal shadowing model was adopted in Kuruvila *et al.* (2005, 2006). It is represented as a function $P(q, x)$, which has approximation within 5% accuracy of the actual one, where q depends on the length L of the considered packet and x is the distance between two nodes. The function is $P(q,x) = 1 - (x/R)^{q\beta}/2$ for $x < R$ and $P(q,x) = (2 - x/R)^{q\beta}/2$ for $R \leq \text{x} < 2R$, and 0 otherwise, where β is the power attenuation factor, which is normally between 2 and 6, and R is determined so that $P(q, x) = 0.5$. Bit reception probability is $P(1, x)$ while the packet reception probability for packets with $L = 120$ bits long is $P(2, x)$.

The *aEPR-u* (expected progress routing) (Kuruvila *et al.*, 2005), nodes send at most u acknowledgements after receiving routing packets. It is to maximize the expected progress made by forwarding. In the example in Figure 4.4, C is the current node, A is a neighbor of C and D is the destination. Let $|CD| = c$, $|AD| = a$, and $|CA| = x$. If all packets can be received successfully, C will forward the packet to A such that $|c-a|$ is maximized. In physical layer model, the probability that A receives the packet from C is $p(x)$ (definition of $p(x)$ has been introduced in Section 1.5 in Chapter 1). The total expected hop count (number of packets) between two nodes at distance x is $f(u, x) = 1/[p(x)(1-(1-p(x))^u)] + u/[(1-(1-p(x))^u)]$ (see Chapter 1). The *aEPR-u* (expected progress routing with acknowledgements) selects the neighbor that maximizes $(c-a)/f(u, x)$.

The localized protocols in Kuruvila *et al.* (2006) do not assume the hop by hop acknowledgements. It was pointed out that the packet delivery rate approaches 1 if a large number of intermediate nodes are placed between the source and the destination and the distance between adjacent nodes approaches 0. On the basis of the observation, the end-to-end routing (*EER*) localized routing simply forwards the packet to neighbor A that maximizes $p(x)$, that is, the neighbor closest to C among neighbors that are closer to D than C. The process continues until the destination is reached or a node cannot find neighbors closer to the destination than itself. Different from EER, the *nEPR* (expected progress routing without acknowledgements) algorithm forwards the packet to a neighbor that maximizes the expected progress $p(x)(c-a)$. Only the neighbors closer to the destination than the current node are considered.

The *InEPR* (iterative expected progress routing) algorithm (Kuruvila *et al.*, 2006) is an improved variant of *nEPR*. The algorithm operates as follows. Similar to *nEPR*, the current node C first finds a neighbor A that maximizes $p(|CA|)(|CD|-|AD|)$. For all common neighbors of A and C, C finds the node B such that $p(|\mathrm{CB}|) \times p(|\mathrm{BA}|) > p(|\mathrm{CA}|)$ and $p(|\mathrm{CB}|) \times p(|\mathrm{BA}|)$ is maximized. Only the neighbors that are closer to the destination than C are considered. The process repeats iteratively by checking the neighbors one by one until no improvement is possible. Node C finally forwards the packet to the selected neighbor B, which will apply the same process for its own forwarding. This algorithm can be generalized by finding shortest weighted path $C, B_1, B_2, \ldots, B_n, A$ toward A. To apply Dijkstra's shortest weighted path algorithm, logarithm of the product of probabilities is applied: $\log(p(AB_1)p(B_1B_2)\ldots p(B_nA)) = \log(p(AB_1)) + \cdots + \log(p(B_nA))$.

Projection progress-based algorithms (Kuruvila *et al.*, 2006) differ from *nEPR* schemes in the progress measure only. The progress is measured by the dot product $CD \cdot CA$ instead of $c-a$, where $CD \cdot CA$ is the dot product of two vectors. The current node C forwards the packet to a neighbor A, which is closer to the destination than itself, such that $p(|CA|)(CD \cdot CA)$ is maximized. The iterative projection progress scheme (Kuruvila *et al.*, 2006) is the same as the *InEPR* except that the projection progress method is employed to find the first candidate node.

The case of variable packet lengths on each hop and routing with hop by hop acknowledgments was studied by Stojmenovic *et al.* (2005). The localized algorithms use the expected number of transmitted bits (energy consumption) instead of the expected hop count in terms of packets in *aEPR-u* and *InEPR*. The expected hop count $f(u, x)$ in *aEPR-u* and *InEPR* is replaced by the expected bit count $g(b, k)$ for routing with acknowledgments. The case of variable packet length and routing without hop by hop acknowledgments was also considered in Stojmenovic *et al.* (2005).

The algorithms described so far are physical layer-based solutions for greedy position-based routing. The recovery procedure of delivery guaranteed routing (Bose *et al*., 1999) can be adapted to the physical layer model. Face routing is based on a planar graph in which edges are normally short. Thus, those edges in the planar graph have relatively high reception probabilities in physical layer model. Therefore, the recovery mode for the physical layer impact routing may proceed in a similar manner as in the UDG model. For each visited node on the face, the shortest cost path to the node can be calculated based on the cost metric, which could take packet reception probability into consideration.

It was pointed out (Frey *et al*., 2005) that BLR can also be adapted to the physical layer by modifying the criterion for selecting the best forwarding neighbor and the appropriate time-out. A given node announces the request for forwarding the packet several times so that the best forwarding neighbor receives it. Similarly, the best forwarding neighbor responds a few times to make sure the response is received and it is selected. The number of duplicated packets depends on the detailed physical layer model.

REFERENCES

Bachrach J, Taylor C. “Localization in sensor networks”. In: Stojmenovic I, editor. Handbook of sensor networks: algorithms and architectures. Wiley; 2005. pp. 277–310.

Barriere L, Fraigniaud P, Narajanan L, Opatrny J. “Robust position-based routing in wireless Ad hoc networks with irregular transmission ranges”. Wireless Commun Mobile Comput 2001;3:141–153.

Blum BM, He T, Son S, Stankovic JA. “IGF: a state-free robust communication protocol for wireless sensor networks”. Technical Report CS-2003-11. Department of Computer Science, University of Virginia; 2003.

Bose P, Morin P, Stojmenovic I, Urrutia J. “Routing with guaranteed delivery in Ad hoc wireless networks”. Proceedings of 3rd ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL M99); 1999. pp. 48–55.

Bose P, Morin P, Stojmenovic I, Urrutia J. “Routing with guaranteed delivery in Ad hoc wireless networks”. ACM Wireless Netw 2001;7(6):609–616.

Caruso A, Chessa S, De S, Urpi A. “GPS-free coordinate assignment and routing in wireless sensor networks”. Proceedings of the 24th Annual Conference on Computer Communications (INFOCOM 2005), Volume 1; 2005. pp. 150–160.

Chavez E, Mitton N, Tejeda H. “Routing in wireless networks with position trees”. Proceedings of International Conference on AD-HOC Networks & Wireless (Ad Hoc Now’07); Morelia, Mexico; 2007 Sept.

Datta S, Stojmenovic I, Wu J. “Internal node and shortcut based routing with guaranteed delivery in wireless networks”. Cluster Comput 2002;5(2):169–178.

Ding J, Sivalingam KM, Kashyapa R, Chuan LJ. “A multi-layered architecture and protocols for large-scale wireless sensor networks”. Proceedings IEEE Vehicular Technology Conference (VCT 2003); Orlando, USA; 2003 Oct.

Elhafsi EH, Mitton N, Simplot-Ryl D. “EtE: end-to-end energy efficient geographic path discovery with guaranteed delivery in ad hoc and sensor networks”. Proceedings of IEEE PIMRC 2008; 2008. To appear.

FINN GG. “Routing and addressing problems in large metropolitan-scale internet-works”. Technical Report ISI/RR-87-180. Information Sciences Institute (ISI); 1987.

FREY H, STOJMENOVIC I. “Geographic and energy-aware routing in sensor networks”. In: STOJMENOVIC I, editor. Handbook of sensor networks: algorithms and architecture. Wiley; 2005. pp. 381–415.

FREY H, STOJMENOVIC I. “On delivery guarantees of face and combined greedy-face routing algorithms in Ad hoc and sensor networks”. Proceedings of the 12th ACM Annual International Conference on Mobile Computing and Networking MOBICOM; Los Angeles; 2006, Sept 23–29. pp. 390–401.

FÜßLER H, WIDMER J, KASEMANN M, MAUVE M, HARTENSTEIN H. “Contention-based forwarding for mobile Ad-hoc networks”. Ad Hoc Netw 2003;1(4):351–369.

GABRIEL KR, SOKAL RR. “A new statistical approach to geographic variation analysis”. Syst Zool 1969;18:259–278.

GOLDENBERG DK, LIN J, MORSE AS. “Towards mobility as a network control primitive”. Proceedings of the Fifth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2004); 2004; 163–174.

GUNGOR VC, SASTRY C, SONG Z, INTEGLIA R. “Resource-aware and link-quality-based routing metric for wireless sensor and actor networks”. Proceedings of the IEEE International Conference on Communications 2007 (ICC 2007); Glasgow, Scotland; 2007. pp. 3364–3369.

HANG C, DAI F, WU J. “On-demand location-aided QoS routing in Ad hoc networks”. Proceedings of International Conference on Parallel Processing (ICPP 2004), Volume 1; 2004. pp. 502–509.

HEISSENBUTTEL M, BRAUN T. “BLR: beacon-less routing algorithm for mobile Ad-hoc networks”. Comput Commun 2004;27(11):1076–1086.

HOFMANN-WELLENHOF B, LICHTENEGGER H, COLLINS J. Global positioning system: theory and practice. 4th ed. Heidelberg: Springer; 1997.

HOU TC, LI VOK. “Transmission range control in multihop packet radio networks”. IEEE Trans Commun 1986;34(1):38–44.

HOU J, LI N, STOJMENOVIC I. “Topology construction and maintenance in wireless sensor networks”. In: STOJMENOVIC I, editor. Handbook of sensor networks: algorithms and architectures: Wiley; 2005. pp. 311–341.

HUANG Q, LU C, ROMAN GC. “Reliable mobicast via face-aware routing”. Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM 2004), Volume 3; 2004. pp. 2108–2118.

INTANAGONWIWAT C, GOVINDAN R, ESTRIN D. “Directed diffusion: a scalable and robust communication paradigm for sensor networks”. Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM’00); 2000. pp. 56–67.

JAIN R, PURI A, SENGUPTA R. “Geographical routing using partial information for wireless Ad hoc networks”. IEEE Pers Commun 2001;8(1):48–57.

JOHNSON D, MALTZ D. “Dynamic source routing in Ad hoc wireless networks”. In: IMIELINSKI T, KORTH H, editors. Mobile computing. Kluwer Academic Publishers; 1996. pp. 153–181.

KALOSHA H, NAYAK A, RUEHRUP S, STOJMENOVIC I. “Select and protest based beaconless georouting with guaranteed delivery in wireless sensor networks”. Proceedings of the 27th Conference on Computer Communications (INFOCOM 2008); Phoenix, Arizona; 2008; pp. 346–350.

KARP B, KUNG HT. “GPSR: greedy perimeter stateless routing for wireless networks”. Proceedings of the 6th ACM/IEEE Annual International Conference on Mobile Computing and Networking (MobiCom-00); 2000. pp. 243–254.

KIM YJ, GOVINDAN R, KARP B, SHENKER S. “On the pitfalls of geographic face routing”. Proceedings of the ACM DIALM-POMC Joint Workshop on Foundations of Mobile Computing. ACM Press; 2005. pp. 34–43.

Kranakis E, Singh H, Urrutia J. "Compass routing on geometric networks". Proceedings of the 11th Canadian Conference on Computational Geometry (CCCG'99). 1999. pp. 51–54.

Kruskal JB. "On the shortest spanning subtree of A Graph and the traveling salesman problem". Proc Am Math Soc 1956;7(1):48–50.

Kuhn F, Wattenhofer R, Zhang Y, Zollinger A. "Geometric Ad-hoc routing: of theory and practice". Proceedings of the 22nd ACM International Symposium on the Principles of Distributed Computing (PODC); 2003a. pp. 63–72.

Kuhn F, Wattenhofer R, Zhang Y, Zollinger A. "Worst-case optimal and average-case efficient geometric Ad-hoc routing". Proceedings of the 4th ACM International Symposium on Mobile Computing and Networking (MobiHoc 2003); 2003b. pp. 267–278.

Kuhn F, Wattenhofer R, Zollinger A. "Asymptotically optimal geometric mobile Ad-hoc routing". In Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (Dial-M'02); 2002; pp. 24–33.

Kuhn F, Wattenhofer R, Zollinger A. "An algorithmic approach to geographic routing in Ad hoc and sensor networks". IEEE/ACM Trans Network 2008;16(1):51–62.

Kuruvila J, Nayak A, Stojmenovic I. "Progress based localized power and cost aware routing algorithms for Ad hoc and sensor wireless networks". Proceedings of the 3rd International Conference on Ad-Hoc Networks and Wireless (ADHOC-NOW 2004), LNCS 3158; 2004. pp. 294–299.

Kuruvila J, Nayak A, Stojmenovic I. "Hop count optimal position based packet routing algorithms for Ad hoc wireless networks with a realistic physical layer". IEEE J Sel Areas Commun 2005;23(6):1267–1275.

Kuruvila J, Nayak A, Stojmenovic I. "Greedy localized routing for maximizing probability of delivery in wireless Ad hoc networks with a realistic physical layer". J Parallel Distrib Comput 2006;66(4):499–506.

Lee S, Bhattacharjee B, Banerjee S. "Efficient geographic routing in multihop wireless networks". Proceedings of the Sixth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2005); 2005. pp. 230–241.

Leong B, Mitra S, Liskov B. "Path vector face routing: geographic routing with local face information". Proceedings of the 13th IEEE International Conference on Network Protocols (ICNP 2005); 2005.

Li XY, Wan PJ, Wang Y. "Power efficient and sparse spanner for wireless Ad hoc networks". Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN); 2001. pp. 564–567.

Lin X, Lakshdisi M, Stojmenovic I. "Location based localized alternate, disjoint, multi-path and component routing schemes for wireless networks". Proceedings of the 2001 ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001); 2001. pp. 287–290.

Liu H, Nayak A, Stojmenovic I. "Localized mobility control routing in robotic sensor wireless networks". Proceedings of the 3rd International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2007), LNCS 4864; 2007; pp. 19–31.

Ma X, Sun M-T, Zhao G, Liu X. "An efficient path pruning algorithm for geographical routing in wireless networks". IEEE Trans Vehicular Technol 2008;57(4):2474–2488.

Matula DW, Sokal RR. "Properties of gabriel graphs relevant to geographic variation research and the clustering of points in the plane". Geogr Anal 1980;12:205–222.

Mitton N, Razafindralambo T, Simplot-Ryl D, Stojmenovic I. "Hector is an energy efficient tree-based optimized routing protocol for wireless networks". The 4th International Conference on Mobile Ad-hoc and Sensor Networks (MSN'08); Wuhan, China; 2008 Dec 10–12.

Perkins C, Royer EM, Das SR. Ad Hoc on Demand Distance Vector (AODV) Routing. Internet Draft, draft-ietf-manet-aodv-04.txt, Oct 1999.

Sanchez JA, Ruiz PM. Locally Optimal Source Routing for Energy-Efficient Geographic Routing. MSN 2006; Wireless Network (WINET). To appear.

Sanchez JA, Ruiz PM, Stojmenovic I. “Energy efficient geographic multicast routing for sensor and actuator networks”. Comput Commun (Elsevier) 2007;30(13):2519–2531.

Shah GA, Bozyiqit M, Aksoy D. “RAT: routing by adaptive targeting in wireless sensor/actor networks”. Proceedings of the Second International Conference on Communication Systems Software and Middleware; 2007. pp. 1–9.

Souryal MR, Moayeri N. “Channel-adaptive relaying in mobile ad hoc networks with fading sensor and Ad hoc communications and networks”. Proceedings of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2005); 2005. pp. 142–152.

Stojmenovic I. Dynamic load balancing for georouting. In preparation.

Stojmenovic I. “Location updates for efficient routing in wireless networks”. Handbook of wireless networks and mobile computing. John Wiley & Sons; 2002. pp. 451–471.

Stojmenovic I. “Localized network layer protocols in sensor networks based on optimizing cost over progress ratio”. IEEE Netw 2006;20(1):21–27.

Stojmenovic I, Datta S. “Power and cost aware localized routing with guaranteed delivery in unit graph based ad hoc networks”. Wireless Commun Mobile Comput 2004;4(2):175–188.

Stojmenovic I, Lin X. “Loop-free hybrid single-path / flooding routing algorithms with guaranteed delivery for wireless networks”. IEEE Trans Parallel Distrib Syst 2001a;12(10):1023–1032.

Stojmenovic I, Lin X. “Power aware localized routing in Ad hoc networks”. IEEE Trans Parallel Distrib Syst 2001b;12(10):1023–1032.

Stojmenovic I, Nayak A, Kuruvila J. “Design guidelines for routing protocols in Ad hoc and sensor networks with a realistic physical layer”. IEEE Commun Mag 2005;43(3):101–106.

Stojmenovic I, Russell M, Vukojevic B. “Depth first search and location based localized routing and QoS routing in wireless networks”. Comput Inform 2002;21(2):149–165.

Takagi H, Kleinrock L. “Optimal transmission ranges for randomly distributed packet radio terminals”. IEEE Trans Commun 1984;32(3):246–257.

Toussaint G. “The relative neighborhood graph of a finite planar set”. Pattern Recognit 1980;12(4):261–268.

Vukojevic B, Goel N, Kalaichevan K, Nayak A, Stojmenovic I. “Depth first search based and power aware geo-routing in ad hoc and sensor wireless networks”. Int J Auton Adaptive Commun Syst (IJAACS) 2008;1(1):41–54.

Wu S, Candan KS. “Power-aware single- and multipath geographic routing in sensor networks”. Ad Hoc Netw 2007;5:974–997.

Zorzi M. “A new contension-based MAC protocol for geographic forwarding in Ad hoc and sensor networks”. Proceedings of the IEEE International Conference on Communications (ICC 2004), Volume 16; 2004; pp. 3481–3485.

Zorzi M, Armaroli A. “Advancement optimization in multihop wireless networks”. Proceedings of the IEEE 58th Vehicular Technology Conference (VTC 2003), Volume 5; 2003; pp. 2891–2894.

Zuniga Zamalloa M, Seada K, Krishnamachari B, Helmy A. “Efficient geographic routing over lossy links in wireless sensor networks”. ACM Trans Sens Netw 2008;4(3), Article 12.

Chapter 5

Multicasting, Geocasting, and Anycasting in Sensor and Actuator Networks

Arnaud Casteigts, Amiya Nayak, and Ivan Stojmenovic

School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, Canada K1N6N5

Abstract

This chapter reviews the scenarios where a given message is sent from a single source to possibly several destinations. These scenarios can be subdivided into *multicasting*, *geocasting*, *multiratecasting*, and *anycasting*. In multicasting, a given message must be routed from one node to a number of destinations whose locations may be arbitrary and spread over the network. Geocasting destinations are all nodes located in a given geographical area. Multiratecasting is a generalization of multicasting, where regular messages are sent from a source to several destinations, possibly at a different rate for each destination. Finally, in an anycasting scenario, a source must send a message to any node among a given set of destinations, preferably only one. Each of these scenarios corresponds to a typical use case in sensor and actuator networks.

5.1 MULTICASTING

In a multicasting task, the same message is routed from one single source node to a fixed number of destinations whose locations are potentially scattered in the network. In the context of sensor and actuator networks, this routing scheme is usually applied by sensors to report their data to several actuators (as illustrated in Fig. 5.1).

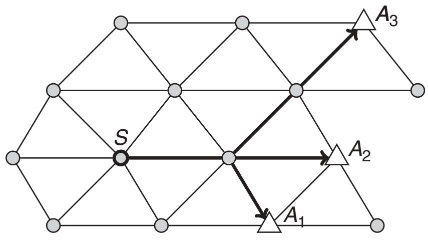


Figure 5.1 Multicasting from a sensor S to actuators A_1, A_2, A_3.

More formally, given a graph $G = (V, E)$, a source $s \in V$, and a set of destinations $D \subseteq V$, the multicast problem consists in finding a set of relay nodes $R \subset V$, such that $s \cup F \cup R$ is connected in G. The main idea behind multicast is to try to reduce R as much as possible (share a maximum of the links to send as few duplicate packets as possible). In most of the algorithms, this path sharing consists in building an overlay tree whose root is the source and leaves are destinations (destinations may also act as relay nodes, though). This is however not always the case, some algorithms rather build a mesh overlay, which better tolerate the failure of links, thanks to redundancy. Also, when the positions of destinations are known beforehand, the multicast task can be achieved without using any overlay structure. In this latter case, the paths may also form trees or meshes, but these structures are built on the fly and not memorized. The present section reviews these different solutions.

5.1.1 Nongeographic Multicast

A number of multicast protocols [DVMRP (Deering and Cheriton, 1990), MOSPF (Moy, 1994), CBT (Ballardie *et al.*, 1993), PIM (Deering *et al.*, 1994)] were first proposed in the context of the Internet, and more generally IP networks. These protocols were specifically designed for wired and infrastructured networks and are not relevant in the context of wireless sensor networks. The main reason is that they do not make use of the broadcast nature of the wireless medium. Indeed, in a wireless context, the transmission of a message from one node to any number of its neighbors can be achieved in one single emission. This property, called the *wireless multicast advantage*, implies to modify both the routing strategy and the metrics used to measure its efficiency: summing the total *number of hops* (edges) does not precisely reflect the efficiency of a wireless path, whereas counting the *number of transmissions* does it better. Another important concern, which is not addressed by the aforementioned protocols, is the one of minimizing the energy consumption. This criterion is not very relevant in an infrastructured network, but it is of utmost importance in networks where devices are self-powered, such as sensor and actuator networks.

A number of multicast protocols taking into account the wireless multicast advantage have been proposed this past decade in the area of mobile *ad hoc* networks (MANETs). These protocols are usually classified according to two criteria: whether they are *proactive* or *reactive*, and whether they rely on *trees* or *meshes*. Proactive protocols compute and maintain routing tables ahead of time, whereas reactive (also called *on-demand*) protocols establish the given route once explicitly needed, and do not maintain it afterward. The reactive approach generally performs better than the proactive in dynamic topologies, since maintaining proactive routing information in this context is expensive. Regarding the shape of the routing structures, tree-based protocols build a tree from the source to the destinations, while mesh-based approaches intentionally add redundancy by considering additional links in order to tolerate topological changes. Meshes are generally considered more robust than trees, but induce a higher overhead.

Examples of tree-based schemes are AMRIS (Wu and Tay, 1999) (which proactively builds a shared multicast tree among a set of sources and destinations), multicast *ad hoc* on-demand distance vector (MAODV) (Royer and Perkins, 2000) [which extends the well-known *ad hoc* on-demand distance vector (AODV) reactive protocol by adding an activable multicast mode on top of paths built by AODV], and adaptive demand-driven multicast routing (ADMR) (Jetcheva and Johnson, 2001) (which constructs an overlay multicast tree from each source to its destinations, with parts of the trees being possibly shared by different sources). Examples of mesh-based approaches include CAMP (Garcia-Luna-Aceves and Madruga, 1999) (an extension of the proactive *core-based trees* protocol that adds redundant links) or ODMRP (Lee *et al.*, 1999) (a reactive protocol that computes several paths among sources and destinations). Some protocols, such as AMROUTE (Xie *et al.*, 2002), combine tree- and mesh-based approaches. Finally, some protocols such as multicast core-extraction distributed *ad hoc* routing (MCEDAR) (Sinha *et al.*, 1999) or the one in Jaikaeo and Shen (2002) rely on a backbone structure.

While using the wireless multicast advantage correctly, these protocols focus more on finding quick and robust multicast paths than on minimizing the energy consumption. This is mainly due to the fact that building an *energy-efficient* multicast tree consumes additional time, which is critical in highly dynamical contexts, because an energy-efficient tree may no longer exist by the time it is computed. In near-static scenarios such as sensor and actuator networks, however, taking the time to build an energy-efficient path is relevant. The problem of finding a minimum cost multicast tree in a wired network (i.e., without considering the wireless advantage) is known to be NP-complete, even when every link has the same cost [this problem is similar to the *Steiner tree* problem, where the weights of all edges are equal to 1 (Karp *et al.*, 1972)]. In a wireless network, where the purpose is to minimize the number of *retransmissions* instead of the number of hops, this problem is still NP-complete, as proven in Ruiz and Gomez-Skarmeta (2005). Some heuristics have been considered, such as in Ruiz *et al.* (2007). However, as for the protocols described before, the routing strategy considered here rely on an overlay structure that covers the whole network. Constructing and maintaining such structure may be very costly in message overhead and does not scale well in large networks. When the positions of nodes are known, a better and more efficient routing can be achieved.

5.1.2 Geographic Multicasting

The general idea behind *geographic routing* is to use the positions of nodes to achieve efficient routing without requiring the construction and maintenance of global routing structures such as trees and meshes. This comes from the observation that the very measurements of sensors do not usually have a proper meaning unless associated with the corresponding geographical information. Hence, assuming the availability of position information on sensors is not so strong a hypothesis. *Geographic unicast* has been discussed in Chapter 4 of this book. We review in this section the protocols that extend geographic routing to *multicast* scenarios. First, geographical, but nonlocalized protocols, which are not optimal in the context of sensor and actuator networks, are briefly mentioned. Subsequently, some protocols that are both geographical and localized are reviewed.

One of the first nonlocalized geographical multicast protocol, lightweight adaptive multicast (LAM) (Ji and Corson, 1998), still makes use of broadcast messages, and is therefore not practical in large wireless sensor networks. Another protocol from the same authors, DDM (Ji and Corson, 2001), combines several unicast data tables to set up multicast data forwarding. The management of these tables requires additional overhead and makes it limited to small multicast groups. Finally, a protocol proposed in Mizumoto *et al.* (2004) uses position information to build a geographically aware multicast tree that aims at minimizing the number of links. However, as discussed before, minimizing the number of edges (hops) does not exploit the *one-to-many* nature of the wireless medium (wireless multicast advantage). Additionally, this protocol builds a global overlay structure, which is costly to maintain and does not scale in large networks.

The best advantage of geographic routing is certainly to avoid the necessity of building and maintaining a global routing structure, while enabling the use of localized and stateless protocols where all the information needed to route a message is carried inside it and the selection of next relay neighbors is done on the fly at each hop. Such routing requires that every node knows (i) its own position using either global positioning system (GPS)-like positioning service, or virtual coordinates (see Section 4.7, or (Niculescu, 2004)), (ii) the position of its direct neighbors (using a beaconing scheme), and (iii) the positions of the destinations (actuators) to report to, by using a location service (topic discussed in Chapter 8). The multicast protocols reviewed below (GMP, PBM, GMR, HGMR, and HRPM) are such protocols.

GMP (Wu and Candan, 2006) works as follows: at every hop, the current forwarding node builds a *virtual* multicast *Steiner tree*, rooted in itself, whose leaves are the real destinations. This tree is obtained by merging the destinations by pair, creating a virtual node to represent them. The process goes recursively until no reduction is profitable. On the basis of this tree and the positions of local neighbors, the destination set is then possibly split into several groups, and a neighbor is chosen to serve each group at the next hop. To deal with void areas while forwarding to a subgroup of destinations, the protocol proposes to use a normal *unicast* face routing [see Section 4.7 or (Bose *et al.*, 1999)], where the destinations of the group are all replaced by a single point, being the average of their geographic locations. The main disadvantage of GMP lies in the heuristic it uses to build the Steiner tree. Indeed, it calculates the virtual nodes by merging only two nodes at a time, which may lead to wrong position estimations, especially if the destinations are scattered in the network or if they are surrounding the source node.

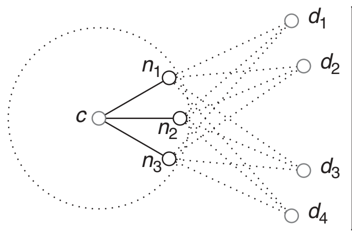
Another protocol, PBM (Mauve *et al.*, 2003), was not initially designed for sensor networks. However, it fulfills the criteria of being localized and using a limited network overhead. This protocol is a generalization of the *greedy-face-greedy* (GFG) principle [see Section 4.5, or (Bose *et al.*, 1999)] to multidestination contexts. It relies on a trade off between individual shortest paths for each destination and overall cost minimization, according to a chosen balance parameter λ. More precisely, in each step, the current forwarding node evaluates each possible combination C of next forwarding neighbors using a function $f(C) = \lambda N + (1 - \lambda)D$, where N is the ratio of selected nodes among the total number of neighbors, D is the sum of all remaining distances from these neighbors to destinations (in proportion to the sum of distances from the current node), and λ is a value between 0 and 1. If the best subset of neighbors is a single node, then that node is the only relay for all the destinations. If a combination with several nodes is selected, then each of these nodes will further take care of a different part of the destinations. When no advance can be provided toward one or more destinations, a variant of *face routing* is used for these destinations, while greedy forwarding continues toward the others. The main problem with this approach is determining the optimal value for λ, as no single value is the best for all scenarios. An additional issue in dense networks or for large multicast groups (large number of destinations) is that the algorithm evaluates all possible association combination between destinations and neighbors (whose complexity grows exponentially with the two factors).

GMR (Sanchez *et al.*, 2007) is another adaptation of GFG to multicast scenarios. The high-level adaptation of GFG is similar to the one of PBM: when a forwarding node cannot find any neighbor providing an advance toward some of the destinations (i.e., a *local optimum* is reached); those destinations are put in a list called the *multicast face list*, and *face routing* is started for them. If a current forwarding node, in face routing, happens to be closer than the previous local optimum for a given destination, then this destination is removed from the face list and added to the greedy list (greedy routing is resumed for this destination). Note that the current node may select a same neighbor to serve in the two modes simultaneously (for different destinations), in which case a single message containing both information is transmitted. While similar to PBM for the high-level strategy, GMR significantly differs concerning the selection of the forwarding neighbors.

In order to solve the complexity problem due to evaluating all the possible combinations, GMR considers an initial default combination, where each destination is individually served by the most convenient neighbor (i.e., the neighbor that provides the highest advance toward it). This leads to a set of destination subsets (or *partitioning*) $P = \{M_1, M_2, \ldots, M_{|P|}\}$, where each subset corresponds to destinations being served by the same neighbor. An example scenario is given in Figure 5.2, where a current node c groups together the destinations d_1 and d_2 (with respect to n_1), and d_3 and d_4 (with respect to n_3). Once the initial partitioning is built, GMR optimizes it by merging the subsets, by pairs, as long as such a merge is possible and profitable. To compare several partitionings, the protocol evaluates them according to the concept of *cost over progress ratio* (see Section 4.3, or (Kuruvila *et al.*, 2005) for general definitions), where *cost* is the number of selected nodes, and *progress* is the overall reduction of remaining distances to destinations (assuming the more appropriate neighbor for each subset). In the example in Figure 5.2, GMR evaluates the initial partitioning P_1, and the partitioning resulting from the merge, P_2. In this scenario, using only n_2 gives a slight distance penalty, while reducing the cost by half. As a result, the merge is profitable and done. In more complex scenarios where several merges are possible, GMR evaluates all of them, applies only the most profitable, then evaluates again.

While GMR solves the main drawbacks of PBM, it still has the disadvantage that the header of every packet contains information about each destination of the packet. Hence, the encoding overhead in each packet is a function of the number of destinations, which become unacceptable if this number grows too large.

HRPM (Das *et al.*, 2008) is a recent protocol that tackles this problem by constructing a hierarchy to serve the destinations. This hierarchy is achieved by geographically dividing the network into cells, where the destinations register and unregister as they move in the network. Thanks to a sophisticated management of these registrations (described later), the source can obtain the list of all the cells that contain at least one destination, and then send the packet to these cells without caring about which particular destination is inside which particular cell. Each cell then forwards the packet to the destinations inside. Actually, a three or four-level hierarchy can be used, but the authors showed that this two-level hierarchy is sufficient to support up to 5800 destinations (with respect to keeping a moderate ratio of header length over data size). The main advantage of this protocol is to guarantee that the per-packet overhead is never more than a desired constant ω. To ensure this property, the dimensions of the cells are determined



Initial partitioning: $P_1 = \{\{d_1, d_2\}, \{d_3, d_4\}\}$, with $\{d_1, d_2\}$ to be served by n_1 and $\{d_3, d_4\}$ by n_3.

Results of the merging process: $P_2 = \{\{d_1, d_2, d_3, d_4\}\}$, with $\{d_1, d_2, d_3, d_4\}$ to be server by n_2.

The two subsets have been merged after comparing cost over progress ratios of P_1 and P_2.

Figure 5.2 Example of routing decision by GMR.

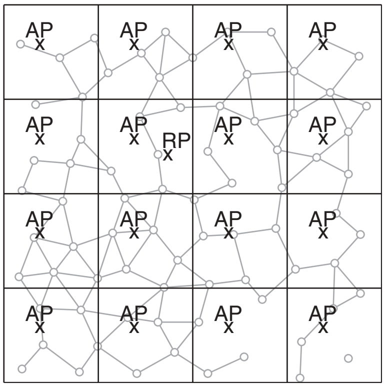


Figure 5.3 Geographical division of the network, as shared by all the nodes of a same multicast group.

according to ω and to the size of the multicast group. As a consequence, each multicast group defines a particular space partitioning and cell management. We describe now, how the cell management works.

The protocol assumes that each multicast group has a unique identifier, and that each potential destination is aware of all the multicast groups it belongs to. This allows all the destinations of the same group to recreate the same local representation of the network division. Thanks to a common *geographic hashing* function, the identifier of each multicast group can be mapped into a particular location in the network, called the *rendezvous point* (RP), and into a particular location in each cell, called the *access point* (AP), as illustrated in Figure 5.3.

For the sake of simplicity, we will first consider that both RPs and APs are real nodes located at the expected locations (this is actually false, but helps the comprehension). Each time a destination moves, it reports its new location to the AP of its cell. When the underlying cell changes, the destination unregisters from one AP, and registers to the other. These memberships are reported by the APs to the RP. When a node wants to send a packet to a given multicast group, it first contacts the RP of this group (thanks to the group identifier and to the geographic hashing function), which sends back the list of cells containing at least one destination. Upon receiving this list, the sending node builds a virtual tree exclusively composed of the corresponding APs, except for the root (itself), and sends the data down the tree (using geographic unicast between each parent and child in the tree). Once the message reaches an AP, another set of geographical unicasts is used to reach the final destinations. The role of the RP, and of each AP, is actually played by the node that is the closest to the corresponding locations, and a special management is locally involved when one of the closest nodes changes. Keeping the AP and RP virtual locations allows to consider them as stationary nodes. For both unicast levels (source to APs, and APs to destinations), the unicast protocol that is used is an adaptation of GFG, which slightly modifies the *face routing mode* to deal with virtual destinations such as the APs or the RP (the modification comes to turn around and select the closest node).

HRPM has several advantages. In addition to the limited encoding overhead, it does not require an external location service, and has a very small group management cost (mainly due to the fact that RPs and APs are stationary locations). However, it is suboptimal regarding the communications. Indeed, it uses a set of unicasts to go down the tree, which may imply at the greedy level to send several times the same message between the same nodes, and does not consider the wireless multicast advantage.

HGMR (Koutsonikolas *et al.*, 2007) is an adaptation of HRPM dedicated to relatively dense and static networks (such as sensor and actuator networks). The general idea behind HGMR is to integrate the design concepts of GMR and HRPM, that is to provide both forwarding efficiency and scalability to large networks. As a recall, transmission of data with HRPM goes from the source to the APs (down a virtual tree of APs), then from each of these APs to the corresponding destinations. GMR has the highest gain when the multicast member density is large (the benefits of broadcasting is maximized), while in sparse networks, its advantage over unicast is mitigated by its encoding overhead (because all destinations are included in the header). Thus, for the transmission from the source to the APs, whose density is expected to be low, HGMR still uses unicast. But within each cell, where destinations are potentially closer from each other, it uses GMR's cost over progress ratio multicast algorithm to select the next relay nodes at each hop. If the number of destinations inside a cell is too large to include all of them in the header, then the geographical decomposition can be adjusted consequently. Hence, the use of GMR within each cell instead of HRPM's unicast-based forwarding strategy helps to reduce the number of transmissions. Another drawback of HRPM was that the hash function could result in the RP being very far from the source. For this reason, HGMR uses a hash function that generates positions within a square in the center of the whole region, which limits the worst cases.

5.2 GEOCASTING WITH GUARANTEED DELIVERY

In a geocasting task, one source node sends a message to all the nodes located in a given geographic region, as illustrated in Figure 5.4. In the context of sensor and actuator networks, this operation is usually applied from an actuator node to all the sensor located in a region of particular sensing interest, this region having possibly different shapes (circular, oval, rectangular, etc.). This message could carry for example, a request for immediate data from any sensor in the region, or inform the sensors about a new location to regularly report to, or give any other instruction that relates to the given region. Geocasting could be used, for example, to monitor the pollution at given locations around a factory (e.g., near the river), or successively requesting the sensors along a moving object trajectory (e.g., tracking the progression of an animal in a forest). Note that geocasting in these cases represents only the request leg, and does not concern the way the data are reported in the reverse direction.

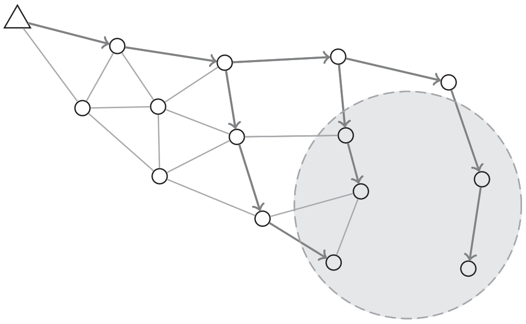


Figure 5.4 Example of geocasting.

As for the other routing principles (e.g., broadcasting, multicasting, or anycasting), a geocasting task can be greatly improved by preliminary putting in sleep mode all the sensor nodes that are not useful to the task, that is, not used to connect the target region, nor to cover it. Such preliminary steps have been discussed in Chapter 3. We present below a review of geocasting protocols that are applicable to wireless sensor and actuator networks. These protocols generally assume that nodes know their own positions and the positions of their neighbors.

5.2.1 Geocasting without Guaranteeing Delivery

For most of the protocols, the task of geocasting consists in two major stages: the first is to reach one node in the targeted geographic region, and the second to inform the other nodes in the region, starting from this node. A simple solution is to route the message from the source to any node in the region using a greedy geographic protocol, and then to use blind flooding (each node inside the region retransmits exactly once) to reach the other nodes, as illustrated Figure 5.5.

Several localized protocols were proposed on this basis (a review of them is available in Stojmenovic and Wu (2006)). Regarding the first stage, all these algorithms are based on a greedy advance, which is restricted to a *virtual area delimitation* between the source (or current node) and the target region, as illustrated by dashed lines in Figure 5.6. Examples of such restricted areas include the space between tangents from the current node to the target region boundaries (*limit 1*), the rectangle containing the source and the target region (*limit 2*), or simply the area offering any physical progression to the target (*limit 3*).

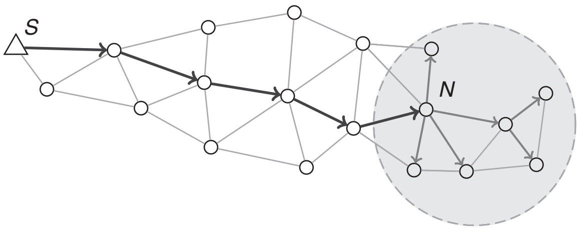


Figure 5.5 Geocasting seen as the combination of unicasting and flooding.

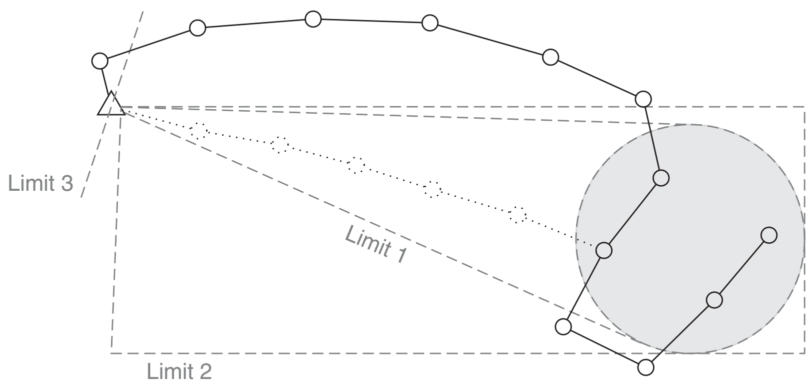


Figure 5.6 Disconnected geocasting region is an obstacle to guarantee the delivery.

These methods inherently do not guarantee the delivery to the region when no route exist within the restricted area. For example, if the dotted nodes in Figure 5.6 were absent from the topology, then no route would be found despite the fact that there exist another one going round above. This problem can be solved by using the *greedy-face-greedy* (GFG) principle to reach the region (see Section 4.5 or Bose *et al.*, 1999). A simple geocasting protocol, proposed in Stojmenovic *et al.* (1999), makes use of GFG to route toward the region, and once inside, performed a flooding within the region. While this algorithm guarantees the delivery to at least one node in the region [under the assumption of an ideal medium access control (MAC) layer], it does not guarantee that all the nodes inside the region will get the message (second stage). Indeed, as shown in Figure 5.6, the sensors that cover a given geocast region may not necessarily be connected inside it, even if the coverage is complete (due to possibly different sensing and communication radii, or obstacles). They can be connected by nodes outside the region, though. The next paragraphs review three geocasting solutions that guarantee the delivery to all the nodes inside the region (provided they are indirectly connected to the source).

5.2.2 Geocasting Based on Traversing Faces that Intersect Boundary

We call *internal (respectively external) border node* a node that is *inside (resp. outside)* the region and has at least one neighbor *outside (resp. inside)* the region in the considered planar subgraph. In Seada and Helmy (2004) an algorithm was proposed that uses the GFG algorithm to forward the packet toward the region, and then activates a perimeter mechanism to guarantee delivery to all nodes inside the region. However, as shown in Stojmenovic and Wu (2006), this mechanism does not actually guarantee delivery, although it is very similar to the following algorithm, that does.

During the first stage, the source node sends the message toward the geocasting region using GFG, which guarantees that the region is reached if connected to the source. Once at the region, three different behaviors are defined for nodes upon reception, depending on which node receives and which node sends.

1. If the receptor node is inside the region, then it retransmits the message in a broadcast fashion. This is done only the first time it receives this message (further copies are ignored). If this node is an internal border node, then it includes in the message an instruction for its external border neighbors, asking them to initiate right-hand face traversals (see case 2 below).
2. If the receptor node is an external border node that receives the message from one of its internal border neighbors, then it initiates a right-hand face traversal to explore the external continuation of each edge that is shared with an internal border neighbor. Further messages coming from any of its internal border neighbors are then ignored.
3. Contrary to the protocol in Seada and Helmy (2004), if the receptor is an external border node and the emitter is a node *outside* the region, then it does not ignore the message (even if it has already performed the step in case 2), and forwards the message along the same face as received. This latter step is necessary to guarantee the delivery in some particular cases.

All these operations are illustrated by the scenario in Figure 5.7. To start, source node S uses GFG to reach the region, which is done at N. According to *case 1*, N initiates a flooding inside the region, and node M initiates a right-hand face traversal along the external continuation of edge NM (note that in the particular

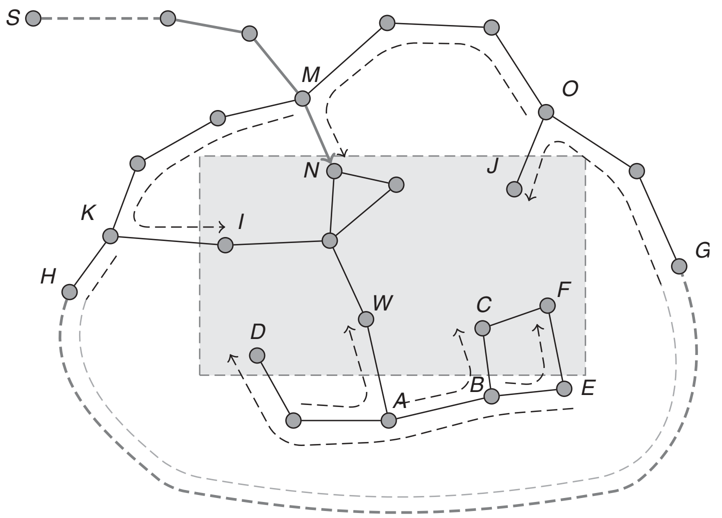


Figure 5.7 Traversing faces that intersect boundaries.

case where the external node gets the message before the internal node, it is not necessary to wait for an instruction in order to start the face traversals). This face traversal reaches I, which ignores the message because it already received it from the inside. Meanwhile, node K initiates the right-hand face traversal continuing IK. This traversal travels around the region (via node H) until coming back at J. Node O initiates a face traversal continuing the edge JO, which closes the loop at N. In the meantime, the first flooding from N has reached node W, which gave an instruction to A to start a face traversal continuing WA. This traversal ends up at node C (which retransmits). Node B starts the face traversal with respect to CB, reaching F (which ignores if already received from C, retransmits otherwise). Finally E starts a face traversal going back to A. At this point, the algorithm from Seada and Helmy (2004) would terminate. Case 3 is applied by A to continue the face traversal and reach node D. A last face traversal closes the loop at W, which ignores the message. In some cases, guaranteeing the delivery implies to visit the whole network. This would have been the case if the dotted round edge between H and G did not map to a real set of connected nodes. Then, the face traversal would have turned around the other side of the network (the reader can imagine the face traversal turning around node H, reaching back M, then passing by the source S, reaching M back again, then traveling until G, turning around it, and finally reaching J).

5.2.3 Geocasting Based on Depth-First Search Traversal of Face Tree

Another geocasting algorithm that guarantees delivery to all the nodes in the region was proposed in Bose *et al.* (2001). Similar to the previous protocol, it does not require any memory to be left at nodes, and needs only to carry some small amount of information with the messages. The protocol consists in applying GFG to route toward a node inside the region, and then to explore all the faces being located inside, or intersecting the region. This exploration, detailed later, is based on the fact that the set of edges that compose any face can be totally ordered by the edges' relative positions to a given reference point in the plane. On the basis of this ordering relation, faces can be associated with one another to form a *face tree* whose root is the face containing the reference point. Then, the region can be entirely visited by performing a depth-first search among this tree of faces. The whole process must be applied on a planar subset of the network (e.g., the Gabriel graph, see Section 4.5).

Given a face f and a point p located outside of the face, the set of edges of f can be *totally ordered* according to their distance to p (actually the distance of the closest point to p in the edge). In case of ties, this total order can be turned *strict* by considering additional comparison keys based on geometrical properties. On the basis of the ordering, each face (except the one containing p) can be associated with one of its own edge, called $entry(f, p)$, being the closest to p. Now, if we consider that each entry edge is on the boundary of two different faces, then these edges define a hierarchy of faces (and thereby an

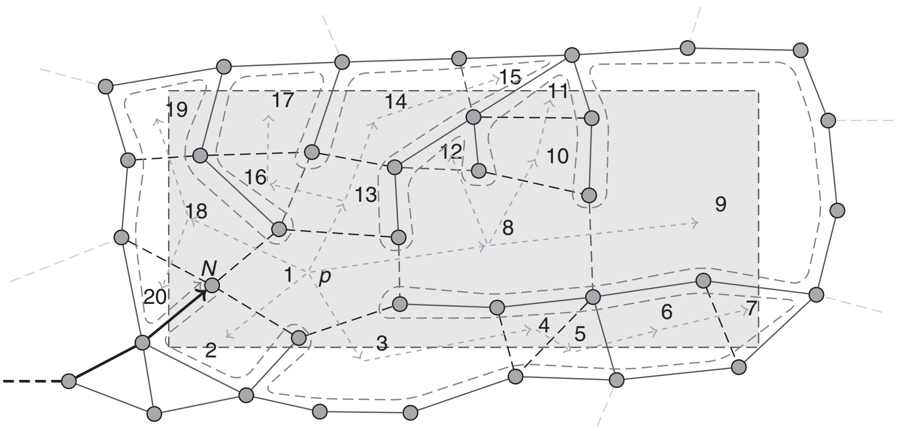


Figure 5.8 Face tree traversal.

implicit tree of faces) such that for each face f except the one containing p, $parent(f, p)$ is the face $f' \neq f$ having $entry(f, p)$ among its edges.

The algorithm works as follows: once the region is reached at one node N by the GFG protocol, this node selects a nearby geographical point p located inside any adjacent faces (which by definition are in the region, or intersect it). This face becomes the root of the face tree. The face tree is then constructed during the geocasting operation, it is in fact the geocasting operation. Starting from the face containing p, at N, the algorithm will visit the entire face, but, before passing each edge, it checks if this edge is an entry edge for the opposite face (the method for this checking is discussed in the next paragraph). If this is the case, then the current face traversal is interrupted in order to visit the child face. This process goes recursively until no child face is found, at which point the traversal of the parent face resumes (at the other end point of the entry edge). In order to limit the visit to the geocasting region, entry edges are defined only for faces that are located in the region, or intersect it. The algorithm is illustrated in Figure 5.8. In this figure, the thick path arriving at N from the outside stands for the first routing stage (e.g., GFG protocol), the continuous dashed line corresponds to the depth-first traversal of the face tree through the whole region, starting at N, and ending at N. The faces are numbered in the sequential order of their visit. Finally, entry edges are represented by dashed black edges, while the corresponding parent/child relations are coded by gray dotted arrows.

Since the algorithm visits all the faces of the face tree (which is composed of all the faces intersecting the region, or being inside), it must necessarily visit all the nodes in these faces, and consequently all the nodes in the region. However, the algorithm suffers a hidden cost: in order to know whether a given edge is an entry edge for the opposite face, that face must be visited. This gives the algorithm a considerable message overhead. One possibility to mitigate this problem would be to determine entry edges only once for all the networks, at the beginning, and memorize them on their end point nodes (assuming no topological changes afterwards). This implies that a common reference point p is agreed at the time of deployment, and that the algorithm is modified to deal with geocasting regions that possibly do not contain p. Let us assume a first step where the message is routed from the geocasting source to any node inside the region (if the source is not already inside it); let s' be this node. If p is inside the region, then geocasting may proceed by backtracking from s' to p using parent links, and then running the normal depth-first search algorithm from p. If p is outside the region, then the algorithm can backtrack from s' toward p, but stops as soon as a node outside the region is reached, say p'. From p', a face traversal is made along the edges outside the region, which corresponds in Figure 5.8 to the external parts of the dashed ride. Along this perimeter, at each *entry* edge (they are known) begins a separate depth-first search traversal, which together visits every face in the region.

As with the original version, this algorithm works only if the geocasting region is convex. Assume for example, a crescent ‘C’ shape with p located near one of the two ends of the shape. The faces at the other end may have parent faces that are outside the region, and thereby may not be visited. This problem could be solved by a similar perimeter traversal, where all entry edges will be detected and induce a separate depth-first search.

Another general optimization for this algorithm could be to work on a well chosen subset of the nodes, namely, a *connected dominating set* (CDS, see Chapter 2 for details). More precisely, we could first find a CDS among the nodes in the region, and then reduce the face tree traversal to these nodes only. By definition, every non-CDS node is at a distance 1 from a CDS node. As a consequence, it is sufficient that only the CDS nodes retransmit in order to reach all the nodes. A face traversal scheme based on CDSs was proposed in Datta *et al.* (2002) to optimize the face mode of GFG. However, such optimization has never been proposed in the context of geocasting and face tree traversal. For illustration purposes, Figure 5.9 shows the same topological scenario depicted in 5.8, with the only difference that face tree traversal is performed on a CDS of the nodes inside the region. Here, the face tree is composed of two faces only: the root, containing p, and one child face. The light gray color depicts non-CDS nodes (and their edges), while normal colors depict the CDS nodes (and the edges between them).

5.2.4 Geocasting Based on Multicasting to the Region Entrance Points

Another strategy to guarantee delivery was proposed in Stojmenovic (2004), and is based on the concept of entrance points. An *entrance point* (also called *external border node* in the previous subsection) is a node that is located outside the target region but has at least one of its neighbors inside it. The strategy consists in reducing the geocasting problem to the problem of reaching every

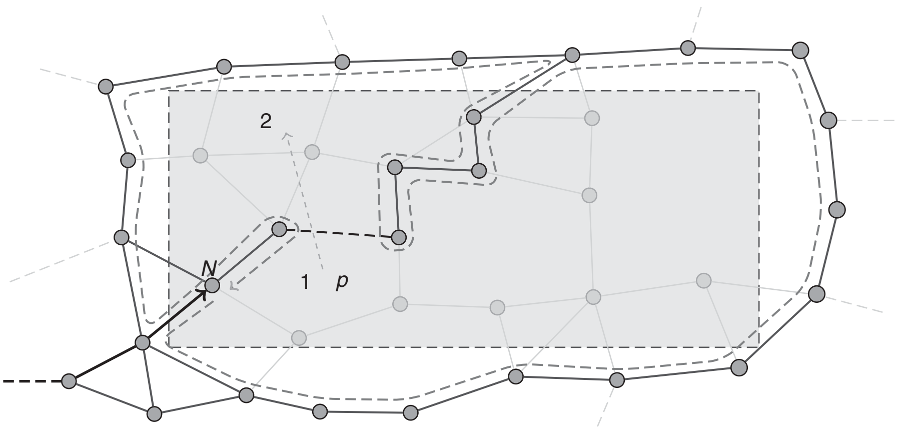


Figure 5.9 CDS-based face tree traversal.

such entrance point, from which intelligent flooding can be initiated to reach the connected nodes inside the region.

Let R be the transmission radius, assumed identical for every node. It can be observed that any entrance point is necessarily (by definition) at a distance $\leq R$ from the region border. It is then possible to reach all entrance points by using geographic multicasting to well chosen areas around the region. These locations, called *entrance zones*, must be determined such that:

1. Any entrance point necessarily belongs to an entrance zone, which implies that the union of all entrance zones must surround the region with a width larger than R.
2. If a node inside any zone retransmits a message, then any other node inside the same zone must receive it. This property ensures the possibility to reach all potential entry points in a zone from any of the nodes in it. The associated requirement is that entrance zones must have a diameter smaller than R.

As illustrated in Figure 5.10, these two constraints cannot be respected together if a single layer of zones is considered (both measures in Fig. 5.10a are incompatible). The exact construction of entrance zones to satisfy these criteria actually depends on the shape of the geocasting region. If the geocasting region is a rectangle, for example, then the entrance zones may be composed of two layers of squares of length $R/2$, or better, composed of rectangles where one dimension (the one perpendicular to the region) is $R/2$ and the other dimension is as large as possible, provided the zone's diameter does not exceed R (as illustrated in Fig. 5.10b). Another example that considers a circular geocasting region is provided in Stojmenovic and Wu (2006).

Once all entrance zones are determined by the source, a geographical multicast task is initiated from the source toward the "centers" of all zones using a

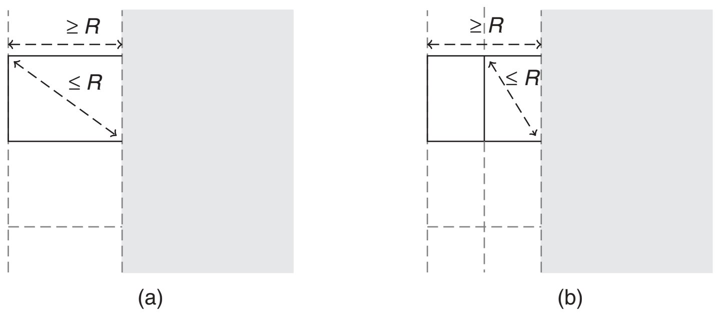


Figure 5.10 Layering of the entrance zones. (a) One-layer entrance zone. (b) Two-layer entrance zone.

protocol such as GMR (see Section 5.1). Note that what is called *center* here can actually be any point inside the zone. Once the multicast is started, the routing paths can split to serve the destinations optimally. For each destination, the process stops when any node inside the corresponding zone is reached, or if a loop in recovery mode is observed (which means that the destination zone is empty or disconnected from the rest of the network). In the case where a node is reached, this node retransmits once in order to reach all the potential entry points located in the same zone, and then these points, if any, initiate an intelligent flooding within the target region.

Some ideas of optimizations for this protocol are proposed in Stojmenovic (2004). For example, several nodes on a path can collectively conclude that a zone (or a set of zones) is empty, and thereby prevent full loops in recovery mode. Another possible optimization is to force nodes to wait for a while between reception and retransmission, during the multicast task, in order to merge the potential routing assignment received by different neighbors. Also, the fact that several flooding operations of the region can be triggered by different entrance zones at different times, requires to adjust time-outs and traffic memorization to somewhat larger values than in regular flooding tasks, in order to ensure that amessage received already, will be recognized and not retransmitted. Note that this time delay may be important due to the possible use of the recovery mode by the multicast protocol.

The main problem of this protocol is, obviously, the overhead induced by empty zones. Indeed, if we do not consider the optimizations discussed above, then each empty zone may trigger a possibly large face traversal within the network, and even if part of the loops can be avoided, the protocol is still expected to incur a large communication overhead in sparse networks. In dense networks, however, this protocol is expected to perform well. Moreover, it has the interesting property of being based on two existing schemes (multicast and flooding), which may reduce the overall memory consumption of routing software on the sensors, if a real world deployment is considered.

In Khan *et al.* (2008), the authors assume scenarios where the geocasting region is always the same. On the basis of this assumption, entrance points to the region can be durably elected (one real node elected to serve each connected component inside the region). The election results are reported to a location server. When a node wants to geocast to the region, it requests the location server, which sends back information about the closest entrance point from the source. Unicast is performed to this closest entrance point, which in turn reaches the others. Every entrance point floods its assigned connected component inside the region.

5.3 RATE-BASED MULTICASTING

Rate-based multicast, or *multiratecasting*, is a generalization of multicasting in which the data sent from a source to the destinations is possibly sent at a different rate for each destination. Let us consider the example of backup base stations in which the sensed data is stored for further analyzes. To ensure fault tolerance, several base stations can be deployed hierarchically, each one collecting the data at a different rate (the highest rate for the primary, a slightly inferior rate for the secondary, etc.). If the primary base station fails, then the secondary base station takes over, and a specific protocol is run to shift the rate requirements among base stations (from the n-ary to the $n + 1$-ary) and to inform the nodes about this change. Finally, the normal report mechanism can be resumed. Other examples may include overlapping sensing area, where the actuators collect data from sensors at a rate inversely proportional to their relative distance. Obviously, these protocols do not generate optimal multiratecast routing paths (which is an NP-complete problem, as generalizing the optimal multicast tree problem in wireless networks has been proven NP-complete in Ruiz and Gomez-Skarmeta (2005)).

The problem of rate-based multicast is very recent. A rate-adaptive multicast protocol has been proposed for mobile *ad hoc* networks in Nguyen *et al.* (2006). This protocol adapts the rate of communication to the quality of the links in order to reduce the overall networking consumption. However, it does not consider the rate as a required parameter, and is therefore not relevant for the problem we are considering. To the best of our knowledge, the only work [prior to Liu *et al.* (2009)] that tackled this problem is in Singh *et al.* (2004). This protocol builds a rate-aware multicast tree by flooding Explore messages from the source to the destinations. Once reached, the destinations send back Ack messages containing their required rates, which build the multicast path on their way back to the source. Some localized techniques are used during this process to optimize the tree. However, the very fact that the protocol uses broadcast and builds a global overlay structure makes it costly and vulnerable to topological changes, thus not well-adapted to the context of sensor and actuator networks. The article by Singh *et al.* (2004) tackled the multiple rate problem, but proposed a nonlocalized protocol that requires a global tree structure to be built. Additionally, the construction of the tree is guided by independent costs for edges, which does not consider the wireless advantage or take into account the rate for the calculation of a path cost.

5.3.1 Rate-Based Metric

As discussed in Liu *et al.* (2009), the *hop count* and *retransmission number* metrics do not reflect the real efficiency of a path in a multiple rate context, as the transmission rate can differ from one relay node to another. This idea is illustrated in Figure 5.11, where two different paths are proposed to serve a given set of destinations (with given rate requirements). Here the shorter path is the most expensive if we consider the number of messages to be effectively sent, that is the *cumulative rate* of retransmissions. Therefore, in order to measure this efficiency, the better choice is to use the sum of the (output) rates at each relay node, which is directly proportional to the number of messages to be sent. More formally, for a given set of relay nodes $R = \{r_1, r_2, \ldots, r_{|R|}\}$ comprising a multicast path, the overall path cost is defined as $\sum_{i=1}^{|R|} rate(r_i)$.

5.3.2 Geographical Rate-Based Multicast

Liu *et al.* (2009) proposed two localized geographical multiratecast protocols: *maximum rate multicast* (MRM) and *optimal rate cost multicast* (ORCM). These two protocols are similar to PBM and GMR (described in Section 5.1) in the sense where they both extend the GFG principle (Bose *et al.*, 1999 or Section 4.5) to the multicast context, and take into account the wireless multicast advantage. The major difference is that they consider the different rates while making decisions, and aim at minimizing the rate-based metric described in the previous paragraph. Regarding assumptions, the protocols assume an ideal MAC layer without loss and that each node is individually capable of forwarding data at the maximum rate among destinations. The routing choices are solely made by looking at local neighbor positions with respect to destination positions; there is no routing table or global overlay structure needed to be built. Finally, the network topology can change between two consecutive routing tasks without other cost than updating the new positions of destinations or the sources, if changed.

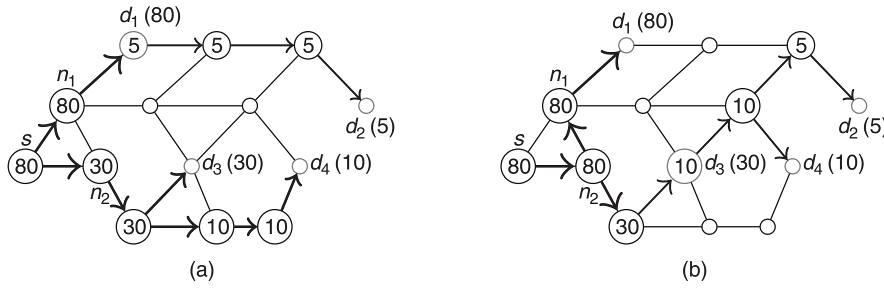


Figure 5.11 Multiratecast to four destinations d_1, d_2, d_3, and d_4. Numbers inside circles indicate the rate at which the corresponding node retransmits. (a) Path A—9 retransmissions, 11 hops, total rate cost: 255. (b) Path B—7 retransmissions, 9 hops, total rate cost: 295.

Both protocols differ only in their greedy mode. The greedy mode is as follows: at each hop, a set of next forwarding nodes is selected by the current node. Each of these nodes is given the responsibility of one or part of the destinations, and will repeat the same selection afterward, until the destinations are reached. The only difference between the two proposed protocols lies in their method to select the right neighbors at each hop. The first protocol, MRM, chooses them linearly by prioritizing the more demanding destinations, while the second, ORCM, evaluates different possible routing choices by combining distance progression and rate considerations, thereby implementing the more general concept of best cost over progress ratio (Kuruvila *et al.*, 2005 or Section 4.3).

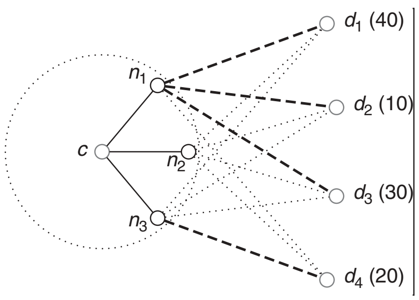
Note that in order to apply such routing, message headers must include, in addition to their positions, the required rate of the destinations.

Maximum Rate Multicast (MRM) Protocol

The basic idea behind MRM is to give priority to destinations that have the highest required rate. More precisely, at each hop, the current node considers the destination that has the highest rate requirement, and determines which neighbor provides the most advance toward it. This neighbor is selected, and all the destinations for which it provides any progression are assigned to it. Then the process is repeated for the remaining destinations, until all of them are assigned. Finally, the message is sent. This process is illustrated in a simple scenario in Figure 5.12.

Optimal Rate Cost Multicast (ORCM) Protocol

Following the examples of PBM and GMR (both discussed in Section 5.1.2 above), the idea behind ORCM is to evaluate different routing choices and then select the best ranked. In order to maintain a moderate calculation complexity, ORCM adapts its strategy to the number of destinations. If this number is under a given threshold, it applies the same strategy as PBM by generating and evaluating all possible combinations (i.e., all possible arrangements of destination subsets, or *partitioning*, such that the destinations of each subset are assigned to a same



The highest rate destination, d_1, is assigned to the neighbor that provides the most advance toward it, n_1. Then all the destinations for which n_1 provide any advance are assigned to it (d_2 and d_3). The process start anew with the remaining destinations, and d_4 is assigned to n_3.

Figure 5.12 Example scenario for MRM.

neighbor). If the number of destinations is above the threshold, then it applies the strategy of GMR, which consists in computing an initial default partitioning (where destinations are grouped according to the most appropriate neighbor to serve them), and then to iterate a merge process to optimize it. The choice for the threshold value basically depends on the expected computational power of the nodes (a threshold of six was considered in Liu *et al.* (2009)).

Let us consider the simple scenario given in Figure 5.13, where the current node c wants to select the next forwarding nodes toward d_1, d_2, d_3, and d_4 (each having possibly a different rate requirement), and has the choice between using only n_1, only n_2, or both for different destinations. We illustrate here, the two high-level strategies, depending on whether the number of destinations (four) is under, or above the threshold.

Strategy 1: evaluation of all set partitions.

There are 14 possible ways of partitioning the destinations:

$$\begin{array}{lll}
P_1 = \{\{d_1, d_2, d_3, d_4\}\} & P_6 = \{\{d_1, d_2\}, \{d_3, d_4\}\} & P_{11} = \{\{d_2\}, \{d_3\}, \{d_1, d_4\}\} \\
P_2 = \{\{d_1\}, \{d_2, d_3, d_4\}\} & P_7 = \{\{d_1, d_3\}, \{d_2, d_4\}\} & P_{12} = \{\{d_2\}, \{d_4\}, \{d_1, d_3\}\} \\
P_3 = \{\{d_2\}, \{d_1, d_3, d_4\}\} & P_8 = \{\{d_1, d_4\}, \{d_2, d_3\}\} & P_{13} = \{\{d_3\}, \{d_4\}, \{d_1, d_2\}\} \\
P_4 = \{\{d_3\}, \{d_1, d_2, d_4\}\} & P_9 = \{\{d_1\}, \{d_2\}, \{d_3, d_4\}\} & P_{14} = \{\{d_1\}, \{d_2\}, \{d_3\}, \{d_4\}\} \\
P_5 = \{\{d_4\}, \{d_1, d_2, d_3\}\} & P_{10} = \{\{d_1\}, \{d_3\}, \{d_2, d_4\}\} &
\end{array}$$

Since there are only two neighbors considered here, the partitions from P_9 to P_{14} do not make sense and can be discarded. ORCM will then evaluate every partitioning from P_1 to P_8, and select the best ranked. On the basis of this best partitioning, each subset will be assigned to a given neighbor (the one minimizing the sum of remaining distances toward the corresponding destinations).

Strategy 2: initial partitioning and merge process. As with GMR, an initial partitioning is generated, where each subset represents the destinations for which the best neighbor is the same, which leads to $P' = \{\{d_1, d_2\}, \{d_3, d_4\}\}$, where n_1 serves $\{d_1, d_2\}$ and n_2 serves $\{d_3, d_4\}$. Then the merge process leads to $P'' = \{\{d_1, d_2, d_3, d_4\}\}$, because it has a better evaluation than P'. All the destinations are then assigned to the same neighbor (the one minimizing the sum of remaining distances toward these destinations).

For the evaluation of a given partitioning, ORCM implements, as GMR, the the concept of *cost over progress ratio*. However, here, the ratio formula is elaborated with the aim of minimizing the new rate-based metric. Three possible

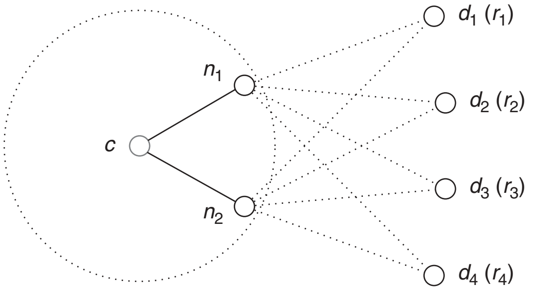


Figure 5.13 Example scenario for ORCM.

methods were proposed in Liu *et al.* (2009) to calculate the cost over progress ratio of a given partitioning, but one of them always outperformed the other two; therefore, we limit the presentation to this one. In order to simplify the formula, the following notations can be introduced:

- Given a set of destinations $D = \{d_1, d_2, \ldots, d_{|D|}\}$, the notation $rate(D) = max(rate(d) : d \in D)$, refers to the highest rate among these destinations.
- Given a current node c, one of its neighbors n, and a destination d, the notation $progress(c, n, d) = dist(c, d) - dist(n, d)$, stands for the progression that n offers from c to d.
- Given a current node c, one of its neighbors n, and a set of destinations $D = \{d_1, d_2, \ldots, d_{|D|}\}$, the notation $progress(c, n, D) = \sum_{i=1}^{|D|} progress(c, n, d_i)$, represents the *cumulative* progress that n offers from c toward these destinations.
- Finally, $N(c)$ stands for the set of neighbors of a node c.

The cost over progress ratio of a given partitioning $P = \{M_1, M_2, \ldots, M_{|P|}\}$, where each M_i is one of the subsets, is defined as the sum of all subset rates, divided by the sum of all maximum subset distance progress. The intuitive idea behind this method is to choose, for each individual subset, the forwarding neighbor that will best profit the whole destination set:

$$ratio(P) = \frac{\sum_{i=1}^{|P|} rate(M_i)}{\sum_{i=1}^{|P|} max(progress(c, n, M_i) : n \in N(c))}.$$

Note that if $max(progress(c, n, M_i : n \in N(c))$ is negative for any of the subsets, then the corresponding partitioning is discarded.

In terms of the new rate-based metric, simulations showed a slight advantage for ORCM over MRM when a small number of destinations are considered (i.e., when ORCM evaluates all possible combinations), and a stronger advantage for MRM otherwise. MRM has also the advantage of a very lower computational cost, even when ORCM does not consider all the destination set partitions. Regarding the absolute efficiency of these protocols, simulations have been run to compare them with a sum of unicast, and have shown an important overall cost advantage for them. Note that as the variance of rate distribution increases, the sum of unicast becomes more efficient than a nonrate-based multicast (which sends the message to all the nodes at the highest rate). An interesting question could be how these protocols behave comparatively to the optimal solution. Answering this question requires designing good approximation algorithms for this NP-complete problem, now.

5.4 ANYCASTING WITH GUARANTEED DELIVERY

In the anycasting problem, a source node wants to send a message to any node that belongs to a given set of destinations. In the context of IP networks, this problem was first formulated in the RFC 1546 as follows: “the host transmits a datagram to an anycast address and the Internetwork is responsible for providing best-effort delivery of the datagram to at least one, and preferably only one” (this was later reformulated in RFC 2373, in the context of IPv6). However, a number of articles refer to the anycasting problem while solving a different problem. For example, in Chen *et al.* (2004) and Jeon and Kesidis, (2007), the term *anycasting* corresponds to the first stage of some geocasting protocols (see Section 5.2), which consists in reaching any node inside a given geographical region.

In the present section, we consider that anycasting typically occurs when a sensor wants to report its data to an actuator, but does not care about which one of them will effectively receive the report. Such a protocol should try to reach the actuator closest to the reported event in order to minimize the energy consumption induced by the report. Finally, actuators must be considered by the protocol as possibly scattered all over the network.

While a number of anycasting protocols were designed for wired networks (Wu *et al.*, 2007), only a few have been designed for wireless networks, and most of them are adaptations of an anycast routing for wired networks (Awerbuch *et al.*, 2003), which rely on flooding techniques. Among other anycasting protocols for wireless sensor networks, in Hu *et al.* (2005), a shortest path anycast tree rooted at each source is constructed for each event source. Sinks are the only leaves of the tree and can dynamically join/leave the tree, which is updated accordingly. Data is delivered to the nearest sink on the tree. The algorithm thus simultaneously maintains paths to all sinks, and requires memorization of routing steps. Algorithms that rely on flooding turn very costly in large networks, and those relying on global structures, such as trees, are very costly to build and maintain as the network becomes larger or dynamic. For these reasons, localized and position-based (geographic) protocols are preferred in the context of sensor and actuator networks, although they require that the positions of actuators are known by the sensor nodes.

Let us introduce a few notations here. They will be used in the following paragraphs. Given two nodes u and v, we denote by $|uv|$ the distance between them. Given a distance d, we denote by $power(d)$ the amount of energy required to transfer a message over this distance. This power is roughly proportional to $d^\alpha + c$, where α represents the *signal strength attenuation factor* and c is a constant factor representing the minimal energy consumption induced by the transfer (see Chapter 1 for more details on energy models). Finally, given a sensor node s, we denote by $A(s)$ the actuator that is the closest to s.

The first position-based anycasting protocol was proposed in Melodia *et al.* (2005). This protocol attempts to minimize the energy consumption as follows. In the startup phase, each sensor node s selects one of its neighbor to act as its next hop toward an actuator. This selection is done by choosing the neighbor n that minimizes $power(|sn|) + power(|nA(n)|)$. Despite the claim of the authors, this localized anycasting algorithm does not really optimize the power consumption because the neighbor selection is based on very long edges $|nA(n)|$, which are not power optimal. Further, the protocol does not guarantee the delivery in presence of void areas.

In Mitton *et al.* (2009), three localized and position-based protocols were proposed. These protocols are three different adaptations of the GFG principle to the context of anycasting, and consequently guarantee the delivery to the destination (i.e., to any actuator). The first protocol, GFGA, attempts to minimize the *number of hops*, while the second, COPA, and the third, EEGDA, focus on minimizing the energy consumption. The next paragraph briefly describes the three selection methods.

1. GFGA selects the neighbor n for which $|nA(n)|$ is minimized ($|sA(s)| - |nA(n)|$ is maximized), in other words it selects the neighbor that provides the highest advance toward any actuator. Note that in this formula $A(s)$ and $A(n)$ can be different, which reflects the key concept of anycasting.
2. COPA relies on the *cost over progress ratio* (Kuruvila *et al.*, 2005, or Section 4.3), where *cost* is the estimated energy consumption for the next hop, and progress is the distance progression toward any actuator. More precisely, for a sensor s, the neighbor n minimizing $power(|sn|)/|sA(s)| - |nA(n)|$ is chosen (two more sophisticated variants are also proposed, but have very similar performance).
3. EEGA is an enhancement of COPA that is inspired from the protocol EtE (Elhafsi *et al.*, 2008). The solution is more energy-efficient than COPA but has a higher computing complexity. The idea behind it is that once a neighbor n is selected as next hop (according to any energy-related cost formula), it may be sometimes more energy-efficient to use one or several additional neighbors to reach n (especially if the signal strength attenuation factor α is high). This is done by calculating the *energy-weighted shortest path* (ESP) from the current node to each neighbor (which can be done locally since nodes are aware of the positions of their neighbors), and then forwarding the message to the first node on the ESP to n.

Regarding the anycasting aspects, the essence of all these solutions is that the destination can be changed during the routing process, to select another actuator if desirable. Such a scenario is illustrated in Figure 5.14, where a sensor node S wants to initiate a report. According to the relative distances of actuators, S selects A_1 as destination, and starts a greedy advance toward it. Once at B, there is no neighbor closer to any actuator than B to A_1. The protocol thus switches to recovery mode (here a right-hand face traversal is started). If the unicast version of GFG were used here, the recovery mode would continue until reaching node D, from which greedy advance would be resumed toward A_1. Here however, node C notices that actuator A_2 is closer to itself than A_1 was to B. A_2 is thus chosen as the new destination, and greedy mode is resumed toward it. Finally, upon receiving the message, node E performs a last modification without breaking the greedy procedure, by switching the destination to A_3, now closer than A_2.

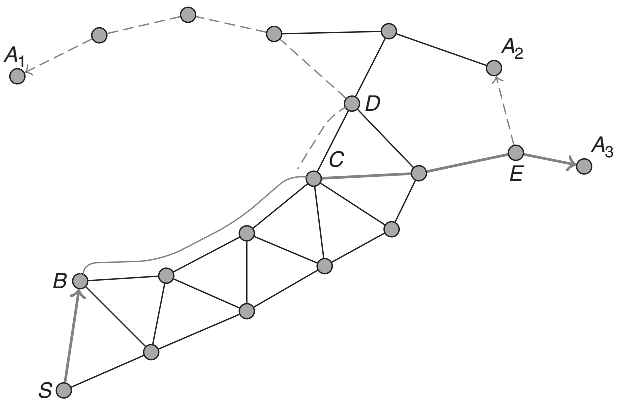


Figure 5.14 Geographic anycasting based on an adaptation of GFG.

REFERENCES

Awerbuch B, Brinkmann A, Scheideler C. “Anycasting in adversarial systems: Routing and admission control”. In 30th International Colloquium on Automata, Languages, and Programming (ICALP); Eindhoven, Netherlands; 2003.

Ballardie T, Francis P, Crowcroft J “Core based trees (CBT)”. SIGCOMM Comput Commun Rev 1993;23(4):85–95.

Bose P, Morin P, Stojmenovic I, Urrutia J. “Routing with guaranteed delivery in Ad hoc wireless networks”. Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM’99); New York, NY, USA; 1999. pp. 48–55.

Bose P, Morin P, Stojmenovic I, Urrutia J. “Routing with guaranteed delivery in Ad hoc wireless networks”. Wirel Netw 2001;7(6):609–616.

Chen S, Dow C, Chen S, Lin J, Hwang S. “An efficient anycasting scheme in ad-hoc wireless networks”. Consumer Communications and Networking Conference; Las Vegas, Nevada, USA; 2004. pp. 178–183.

Das S, Pucha H, Hu YC. “Distributed hashing for scalable multicast in wireless Ad hoc networks”. IEEE Trans Parallel Distrib Syst 2008;19(3):347–362.

Datta S, Stojmenovic I, Wu J. “Internal node and shortcut based routing with guaranteed delivery in wireless networks”. Cluster Comput 2002;5(2):169–178.

Deering SE, Cheriton DR. “Multicast routing in datagram internetworks and extended lans”. ACM Trans Comput Syst 1990;8(2):85–110.

Deering S, Estrin D, Farinacci D, Jacobson V, Liu CG, Wei L. “An architecture for wide-area multicast routing”. SIGCOMM Comput Commun Rev 1994;24(4):126–135.

Elhafsi EH, Mitton N, Simplot-Ryl D. “Energy efficient geographic path discovery with guaranteed delivery in Ad hoc and sensor networks”. IEEE PIMRC; Cannes, France; 2008.

Garcia-Luna-Aceves J, Madruga E. “The core-assisted mesh protocol”. IEEE J Sel Areas Commun 1999;17:1380–1394.

Hu W, Bulusu N, Jha S. “A communication paradigm for hybrid sensor/actuator networks”. J Wirel Inf Netw 2005;12(1):47–59.

Jaikaeo C, Shen C-C. “Adaptive backbone-based multicast for Ad hoc networks”. IEEE International Conference on Communications, ICC, Volume 5; New York, NY, USA; 2002. pp. 3149–3155.

Jeon P, Kesidis G. “Geoppra: an energy efficient geocasting protocol in mobile Ad hoc networks”. International Conference on Networking (ICN); Sainte-Luce, Martinique; 2007.

Jetcheva JG, Johnson DB. “Adaptive demand-driven multicast routing in multi-hop wireless Ad hoc networks”. MobiHoc’01: Proceedings of the 2nd ACM International Symposium on Mobile Ad hoc Networking & Computing. New York: ACM; 2001. pp. 33–44.

Ji L, Corson MS. “A lightweight adaptive multicast algorithm”. Global Telecommunications Conference. GLOBECOM’98. The Bridge to Global Integration. IEEE, Sydney, Australia, Volume 2; 1998. pp. 1036–1042.

Ji L, Corson MS. “Differential destination multicast-a manet multicast routing protocol for small groups”. Proceedings IEEE, INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies; Anchorage, Alaska, USA, Volume 2; 2001. pp. 1192–1201.

Karp RM. “Reducibility among combinatorial problems”. In: Miller RE, Thatcher JW, editors. Complexity of computer computations: New York: Plenum Press; 1972. pp. 85–103.

Khan FA, Ahn K-J, Song W-C. “Geocasting in wireless Ad hoc networks with guaranteed delivery”. Information Networking. Towards Ubiquitous Networking and Services: International Conference, ICOIN 2007; Estoril, Portugal; January 23–25, 2007; 2008. pp. 213–222.

Koutsonikolas D, Das SM, Hu YC, Stojmenovic I. “Hierarchical geographic multicast routing for wireless sensor networks”. Proceedings of International Conference on Sensor Technologies and Applications (SENSORCOMM 2007), Valencia, Spain; October 14–20, 2007.

Kuruvila J, Nayak A, Stojmenovic I. “Hop count optimal position-based packet routing algorithms for Ad hoc wireless networks with a realistic physical layer”. IEEE J Sel Areas Commun 2005;23(6):1267–1275.

Lee S-J, Gerla M, Chiang C-C. “On-demand multicast routing protocol”. Wireless Communications and Networking Conference, 1999. IEEE, WCNC’99; New Orleans, LA, USA, Volume 3; 1999. pp. 1298–1302.

Liu X, Casteigts A, Goel N, Nayak A, Stojmenovic I. “Multiratecast in wireless fault-tolerant sensor and actuator networks”. 2nd IEEE International Conference on Computer Science and its Applications (CSA’09), Jeju, Korea; Dec. 10–12, 2009.

Mauve M, Füssler H, Widmer J, Lang T. “Position-based multicast routing for mobile ad-hoc networks”. SIGMOBILE Mob Comput Commun Rev 2003;7(3):53–55.

Melodia T, Pompili VC, Akyildiz I. “A distributed coordination framework for wireless sensor and actor networks”. ACM Mobihoc; Urbana-Champaign, IL, USA; 2005. pp. 99–110.

Mitton N, Simplot-Ryl D, Stojmenovic I. “Guaranteed delivery for geographical anycasting in wireless multi-sink sensor and sensor-actor networks”. In Proceedings of the 28th Annual IEEE Conference on Computer Communications (INFOCOM 2009); Rio de Janeiro, Brazil; April 2009.

Mizumoto A, Yamaguchi H, Taniguchi K. “Cost-conscious geographic multicast on manet”. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on Sensor and Ad hoc Communications and Networks; Santa Clara, CA, USA; 2004. pp. 44–53.

Moy J. “Multicast routing extensions for ospf”. Commun ACM 1994;37(8):61–ff.

Nguyen UT, Asif A, Xiong X. “Multirate-aware multicast routing in manets”. IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS); Vancouver, Canada; 2006. pp. 554–557.

Niculescu D. “Positioning in Ad hoc sensor networks”. IEEE Netw 2004;18(4):24–29.

Royer E, Perkins C. “Multicast Ad hoc on-demand distance vector (maodv) routing”, 2000.

Ruiz PM, Gomez-Skarmeta AF. “Approximating optimal multicast trees in wireless multihop networks”. IEEE Symposium on Computers and Communications; La Manga del Mar Menor, Spain; 2005. pp. 686–691.

Ruiz PM, Stojmenovic I. "Cost-efficient multicast routing in Ad hoc and sensor networks'. In: Gonzalez T, editor. Volume 65, Handbook on approximation algorithms and metaheuristics. New York: Chapman & Hall/CRC; 2007. pp. 1–14.

Sanchez JA, Ruiz PM, Liu X, Stojmenovic I. "Bandwidth-efficient geographic multicast routing protocol for wireless sensor networks". IEEE Sens J 2007;7(5):627–636.

Seada K, Helmy A. "Efficient geocasting with perfect delivery in wireless networks". IEEE Wireless Communications and Networking Conference, WCNC 2004; Atlanta, Georgia, Volume 4; 2004. pp. 2551–2556.

Singh G, Pujar S, Das S. "Rate-based data propagation in sensor networks". IEEE Wireless Communications and Networking Conference, WCNC 2004, Volume 4; 2004 pp. 2480–2485.

Sinha P, Sivakumar R, Bharghavan V. "Mcedar: multicast core-extraction distributed Ad hoc routing". IEEE Wireless Communications and Networking Conference, WCNC 1999; New Orleans, LA, USA, Volume 3; 1999. pp. 1313–1317.

Stojmenovic I. "Geocasting with guaranteed delivery in sensor networks". IEEE Wirel Commun 2004;11(6):29–37.

Stojmenovic I. "Geocasting in Ad hoc and sensor networks'. In: Wu J, editor. In theoretical and algorithmic aspects of sensor, Ad hoc wireless and peer-to-peer networks. Auerbach Publications (Taylor & Francis Group); 2006. pp. 79–97.

Stojmenovic I, Ruhil AP, Lobiyal DK. Voronoi diagram and convex hull based geocasting and routing in wireless networks. Eighth IEEE Symposium on Computers and Communications, 2003, Antalya, Turkey. Technical Report, SITE, University of Ottawa; 1999.

Wu S, Candan KS. "GMP: distributed geographic multicast routing in wireless sensor networks". International Conference on Distributed Computing Systems; Lisboa, Portugal; 2006. p. 49.

Wu C-J, Hwang R-H, Ho J-M. "A scalable overlay framework for internet anycasting service". Symposium on Applied Computing (SAC); New York; 2007. pp. 193–197.

Wu CW, Tay YC. "Amris: a multicast protocol for Ad hoc wireless networks". IEEE Military Communications Conference Proceedings, MILCOM 1999, Atlantic City, NJ, USA; Volume 1; 1999. pp. 25–29.

Xie J, Talpade RR, Mcauley A, Liu M. "AMRoute: Ad hoc multicast routing protocol". Mob Netw Appl 2002;7(6):429–439.

Chapter 6

Sink Mobility in Wireless Sensor Networks

Xu Li, Amiya Nayak, and Ivan Stojmenovic

School of Information Technology and Engineering, University of Ottawa, Ottawa, Canada K1N6N5

Abstract

Data gathering is a fundamental task of wireless sensor networks (WSNs). It aims to collect sensor readings from sensory fields at predefined *sinks* (without aggregating at intermediate nodes) for analysis and processing. Research has shown that sensors near a data sink deplete their battery power faster than those far apart due to their heavy overhead of relaying messages. Nonuniform energy consumption causes degraded network performance and shortens network lifetime. Recently, *sink mobility* has been exploited to reduce and balance energy expenditure among sensors. The effectiveness has been demonstrated both by theoretical analysis and by experimental study. In this chapter, we investigate the theoretical aspects of the uneven energy depletion phenomenon around a sink, and address the problem of energy-efficient data gathering by mobile sinks. We present a taxonomy and a comprehensive survey of state of the art on the topic.

6.1 INTRODUCTION

Sinks are capable machines with rich (often considered unlimited) resources. Sensors that are generating data are called *sources*. They transmit their data to one or more sinks for analysis and processing. In this chapter, we consider data gathering from sensors, where sensor data are not aggregated on the way to the sink. That is, each sensor measurement arrives at the sink without any changes.

Data transmission could take place either in a push mode or in a pull mode. In the push mode, sources actively send data to sinks; in the pull mode, they transmit only upon sinks' request. The main source-to-sink communication pattern is a multihop message relay, as sinks are out of the transmission ranges of most of sources. The communication paths from reporting sources to a sink form a reverse multicast tree rooted at the sink. Figure 6.1 shows three source-to-sink paths. It is noticed (Ingelrest *et al.*, 2004; Luo and Hubaux, 2005; Olariu and Stojmenovic, 2006; Vincze *et al.*, 2007) that, the closer a sensor is to a sink, the faster its battery exhausts. According to Wadaa *et al.* (2005), Lian *et al.* (2006), and Olariu and Stojmenovic (2006), by the time the one-hop neighboring sensors of a sink deplete their battery power, those farther away may still have more than 90% of their initial energy.

The reason for this phenomenon is intuitively simple: compared with sensors far apart from a sink, sensors that are nearby, are shared by more sensor-to-sink paths, have heavier message relay load, and therefore consume more energy. Researchers have built energy models (Luo and Hubaux, 2005; Lian *et al.*, 2006; Olariu and Stojmenovic, 2006; Banerjee *et al.*, 2009) to provide a formal explanation. Uneven energy depletion causes *energy holes* and leads to degraded network performance. If sensors around a sink all run out of energy, the sink will be isolated from the network; if all sinks are isolated, then the entire network fails. Since manual replacement/recharge of sensor batteries is often infeasible due to operational factors, it is desired to minimize and balance energy usage among sensors.

Power-aware routing (Singh *et al.*, 1998; Stojmenovic and Lin, 2001; Buragohain *et al.*, 2005) has been studied to avoid energy-scarce sensors and achieve longer network lifetime. As indicated in Stojmenovic and Lin (2001), Li and Hou (2004), and Olariu and Stojmenovic (2006), proper use of multilevel transmission radii can balance energy consumption. It was also suggested to use nonuniform node distribution (i.e., the closer to a sink an area is, the higher the node density) to mitigate message relay load and increase network lifetime (Stojmenovic *et al.*, 2005; Lian *et al.*, 2006; Wu *et al.*, 2008). The first

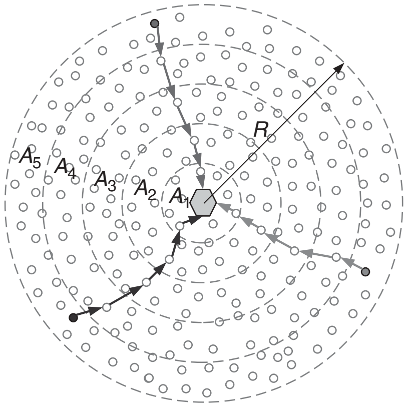


Figure 6.1 Annulus division and sensor-to-sink routing.

two approaches have limited effectiveness since nodes around a sink are very likely to be critical to sink connectivity and cannot be skipped, while the third approach reduces network sensing coverage, which is the functional basis of any sensor network.

Recently, it has been shown (Akkaya *et al.*, 2005; Luo and Hubaux, 2005; Vincze *et al.*, 2007; Banerjee *et al.*, 2009; Basagni *et al.*, 2008; Hashish and Karmouch, 2009; Friedmann and Boukhatem, 2009) that sink mobility can effectively improve network lifetime without bringing above-mentioned negative impacts on the network. The reason is evident: as sinks move, the role of the "hot spot" (i.e., heavily loaded nodes around sinks) rotates among sensors, resulting in balanced energy consumption. In this chapter, we draw attention to the emerging and promising sink mobility problem. We investigate the energy hole problem from the theoretical point of view in Section 6.2. Then, we present a taxonomy of sink mobility approaches for energy-efficient data gathering in Section 6.3. We review existing solutions in Sections 6.4 and 6.5.

6.2 ENERGY HOLE PROBLEM

A wireless sensor network (WSN) with multiple sinks can be divided into subnetworks, each of which is composed of a single sink, data sources reporting to the sink, and sensors relaying messages for the sources. Sensors that appear in more than one such subnetwork will consume energy for all their participating subnetworks. Hence, without loss of generality, we investigate theoretical aspects of the energy hole problem, that is, the uneven energy depletion phenomenon, in single-sink WSNs. We will establish power consumption models for sensor-to-sink communication.

Because exact energy usage prediction is not possible due to network diversity, uncertainty, and dynamics, the models to be presented below are obtained through reasonable approximation. We first present network models and assumptions; then we establish the energy consumption models in two different network scenarios, where sensors have fixed or variable transmission radius respectively. For these two scenarios, we also show how to balance energy usage by applying nonuniform sensor distribution or adjustable transmission radii. The content of this section is based on Olariu and Stojmenovic (2006).

6.2.1 Network Model and Assumptions

Denote by $E_t(d)$ the amount of energy consumed by the sender for transmitting one data bit to distance d, and by E_r the amount of energy spent by receiver in receiving one data bit. The total cost of transmitting one data bit between sender and receiver in one hop is $E_c(d) = E_t(d) + E_r$. We adopt the following general power consumption model (Rodoplu and Meng, 1999): $E_t(d) = ad^\alpha + b$ and $E_r = b$, where $a > 0$ is a constant standing for the transmitter amplifier, $b > 0$ is a constant representing energy for running electronic circuit, and $2 \leq \alpha \leq 6$.

Then we have $E_c(d) = ad^\alpha + 2b$. After normalization, the energy consumption is proportional to

$$E_c(d) = d^\alpha + c, \tag{6.1}$$

where $2 \leq \alpha \leq 6$ and $c > 0$ are constants. For simplicity of analysis, it is assumed that the whole energy consumption is charged to sender node.

Define *node density* as number of nodes per unit area. Sensors are uniformly distributed with density ρ in a circular area of radius R, where a sink is located at the center. Each sensor has a maximum transmission radius r_c that is much smaller than R. There are T sources uniformly scattered in the network and transmitting data to the sink at constant rate λ.

For analysis purpose, we divide the network area into annuli by q concentric circles C_i $(0 \leq i \leq q)$ centered at the sink. Denote by R_i the radius of C_i. We define $R_0 = 0$ and $R_q = R$. Thus, C_0 represents the sink node, while C_q stands for the entire network area. Two adjacent circles C_i and C_{i-1} define the ith annulus for $1 \leq i \leq q$. There are q annuli in total. Denote by A_i the area of the ith annulus and by w_i the width of A_i. We have $A_i = \pi(R_i^2 - R_{i-1}^2)$ and $w_i = R_i - R_{i-1}$. Figure 6.1 illustrates this division method.

We assume that each source is associated with a unique source-to-sink path, which contains exactly one node from each annulus. Further, we assume that each sensor in annulus A_i is equally likely to serve as the next hop for a path that involves a node in A_{i+1}. For simplicity, we assume that the transmission radius needed to send messages between A_i and A_{i-1} is w_i.

6.2.2 Energy Consumption Models

In this section, we are going to establish energy consumption models based on above network model and assumptions. Let n denote the total number of nodes in the network and $A = \pi R^2$ the area of the network field (i.e., the area of C_q). We have

$$n = \rho A = \rho\pi R^2. \tag{6.2}$$

The expected number n_i of nodes in A_i $(1 \leq i \leq q)$ is

$$n_i = \rho A_i = \rho\pi(R_i^2 - R_{i-1}^2). \tag{6.3}$$

For uniform distribution of sources, the expected number T_i of sources in A_i is

$$T_i = T\frac{A_i}{A} = T\frac{R_i^2 - R_{i-1}^2}{R^2}. \tag{6.4}$$

Because source-to-sink paths associated with sources in annuli A_j $(j > i)$ all have the sink as the destination; sensors in A_i collectively participate in all these paths as message forwarders. The expected number $m_{\mathrm{fw}}(i)$ of such paths per node in A_i is

$$m_{\mathrm{fw}}(i) = \frac{1}{n_i} \sum_{i<j\leq q} T_j = \frac{T}{n_i} \sum_{i\leq j\leq q} \frac{R_j^2 - R_{j-1}^2}{R^2} = \frac{T}{n_i} \frac{R^2 - R_i^2}{R^2}. \tag{6.5}$$

The expected number $m_{\mathrm{og}}(i)$ of paths originated per node in A_i is

$$m_{\mathrm{og}}(i) = \frac{T_i}{n_i}. \tag{6.6}$$

Hence, the energy consumption $E(i)$ of each sensor in A_i is

$$E(i) = (m_{\mathrm{fw}}(i) + m_{\mathrm{og}}(i))E_c(w_i).$$

According to Eqs 6.1–6.6, we have

$$E(i) = \frac{\lambda T}{\rho\pi R^2} \frac{w_i^\alpha + c}{R_i^2 - R_{i-1}^2}(R^2 - R_{i-1}^2). \tag{6.7}$$

Equation 6.7 is a general formula describing sensor energy consumption behavior. From this equation, it is not difficult to find that $E(i)$ is proportional to λ and T and reverse proportional to ρ and R^2. When every sensor is a source, that is, when $T = n = \rho\pi R^2$, $E(i)$ becomes independent from T and ρ. When fixed parameters λ, T, R, and ρ are ignored, Eq. 6.7 becomes:

$$E(i) = \frac{w_i^\alpha + c}{R_i^2 - R_{i-1}^2}(R^2 - R_{i-1}^2). \tag{6.8}$$

Let us now determine the optimal w_i that minimizes $E(i)$ for $1 \leq i \leq q$. Note that w_i must not be larger than r_c, because otherwise, the network may be partitioned. We will examine the case that sensors have fixed transmission radius and the case that sensors have adjustable transmission radii, respectively.

Fixed Transmission Radius

When sensors have a fixed communication radius r_c, a node in A_i always has the same power consumption for transmission. In this case, w_i can be replaced with r_c in Eq. 6.7. The optimal w_1 can be determined by examining $E(1) = \frac{r_c^\alpha + c}{R_1^2} R^2$. We observe that, to minimize $E(1)$, R_1 (i.e., w_1) needs to be set to the largest value, r_c. Using this result, we can recursively determine that, to minimize $E(i)$, we should have $R_i = ir_c$ and $w_i = r_c$. Then $R = R_q = qr_c$. From Eq. 6.8 we have the following normalized optimal energy consumption $E_{\mathrm{opt}}(i)$ per node in A_i:

$$E_{\mathrm{opt}}(i) = \frac{r_c^\alpha + c}{2i - 1}(q^2 - (i-1)^2). \tag{6.9}$$

It is seen from Eq. 6.9 that uneven energy depletion occurs around the sink: the closer a sensor is to the data sink, the larger its energy consumption rate is, and thus the faster it depletes its battery power.

Balancing Energy Usage by Nonuniform Node Distribution We will discuss how to balance energy consumption by properly applying different node density in different annuli. Let us denote node density in annulus A_i by ρ_i. It is intuitively clear that in order to balance energy usage an annulus close to the sink should contain more nodes for sharing message relay load than a relatively distant one, namely, $\rho_q < \rho_{q-1} < \cdots < \rho_1$. Our objective is to determine ρ_i as a function of ρ_q such that $E_{\text{opt}}(i) = E_{\text{opt}}(q)$ for $1 \le i \le q$ and $q = R/r_c$.

Replace ρ with ρ_i in Eq. 6.7. Note that $E_{\text{rate}}(i)$ now also depends on ρ_i. By a similar discussion, we obtain normalized optimal energy consumption $E_{\text{opt}}(i)$ per node in A_i:

$$E_{\text{opt}}(i) = \frac{1}{\rho_i}\frac{r_c^\alpha + c}{2i-1}(q^2 - (i-1)^2). \tag{6.10}$$

From $E_{\text{opt}}(i) = E_{\text{opt}}(q)$, we have

$$\frac{1}{\rho_i}\frac{r_c^\alpha + c}{2i-1}(q^2 - (i-1)^2) = \frac{1}{\rho_q}\frac{r_c^\alpha + c}{2q-1}(q^2 - (q-1)^2).$$

Applying simple calculus to above equation, we obtain ρ_i as a function of ρ_q:

$$\rho_i = \rho_q \frac{q^2 - (i-1)^2}{2i-1}. \tag{6.11}$$

Variable Transmission Radius

Now let us assume each sensor is able to adjust its transmission radius up to r_c. Assume that ideally sensors are able to forward along a straight line from source to sink, with transmission radii corresponding to annuli widths. Hence the energy consumption of the route will be

$$E_{\text{path}}(i) = \sum_{j=1}^{i}(w_j^\alpha + c) = \sum_{j=1}^{i} w_j^\alpha + ic. \tag{6.12}$$

By the above equation, $E_{\text{path}}(i)$ is minimized whenever $\sum_{j=1}^{i} w_j^\alpha$ is minimized. Define $a_j = w_j^{\frac{\alpha}{2}}$ for $1 \le j \le i$. Then $\sum_{j=1}^{i} a_j^2 = \sum_{j=1}^{i} w_j^\alpha$. By Lagrange's identity, $\sum_{1\le p\le m\le i}(a_p - a_m)^2 = i\sum_{j=1}^{i} a_j^2 - (\sum_{j=1}^{i} a_j)^2$. Therefore,

$$\sum_{j=1}^{i} w_j^\alpha = \frac{1}{i}\sum_{1\le p\le m\le i}(a_p - a_m)^2 + \frac{1}{i}\left(\sum_{j=1}^{i} a_j\right)^2.$$

We will show that $\sum_{j=1}^{i} w_j^\alpha$ can be minimized by considering each of the expressions on the right side separately, by observing that they are both minimal for the same values. $\sum_{1\le p\le m\le i}(a_p - a_m)^2 = 0$ is an obvious minimal value, which occurs iff $a_q = a_{q-1} = \ldots = a_1$, that is, $w_q = w_{q-1} = \ldots = w_1 = R_1$. It is well known that the power mean function $M(x) = \left(\frac{\sum_{j=1}^{i} w_i^x}{n}\right)^{1/x}$ is a non-decreasing function. Apply it for specific values $x = \alpha/2$ and $x = 1$. The value for $x = 1$ is constant (since the sum of annuli widths is fixed), while the value for $x = \alpha/2$ can be equal to that constant for $w_1 = w_2 = \cdots = w_q$. Note that the proof originally presented in Olariu and Stojmenovic (2006) did not minimize both sums and thus remained incomplete. Hence,

$$R_i = iR_1.$$

We see that the key is to determine R_1. By Eq. 6.8, $E(1) = \frac{R_1^\alpha + c}{R_1^2}R^2$. When $\alpha = 2$, $E(1)$ is minimized for $R_1 = r_c$. Now examine the case of $\alpha > 2$. Given α and c, the value of $R_1 = (\frac{2c}{\alpha-2})^{1/\alpha}$ minimizes $E(1)$ (it is the value for which the derivative of this function is equal to 0). Because sensors' transmission radii are bounded by r_c, we have

$$R_1 = \begin{cases} r_c & \text{for } \alpha = 2 \\ \min\left\{r_c, \left(\dfrac{2c}{\alpha-2}\right)^{1/\alpha}\right\} & \text{for } \alpha > 2. \end{cases} \tag{6.13}$$

Note that the optimal choice for R_1 does not depend on R, the radius of the network area.

Substituting iR_1 for R_i in Eq. 6.8, we obtain the normalized energy consumption per route for a node in A_i as follows:

$$E_{\text{opt}}(i) = \frac{R_1^\alpha + c}{2i - 1}(q^2 - (i-1)^2). \tag{6.14}$$

This is the same expression as Eq. 6.9. Minimizing energy consumption per path leads to higher energy depletion around the sink.

Balancing Energy Usage by Adjusting Transmission Radius We will show how to enable sensors to have the same energy consumption rate (thus balanced energy usage) across the entire disk of radius R by tailoring the annuli widths. It is intuitively clear that, for sensors to have a uniform energy consumption rate, an annulus close to the sink (where message relay load is heavier) must have a smaller width for reducing the sensors' energy usage on cross-annulus transmission than a relatively distant one, namely, the inequality $w_1 < w_2 < \cdots < w_q$ must hold. Our objective is to determine optimal w_1 (i.e., R_1) and then compute w_i as a function of w_1 such that $E(i) = E(1)$.

The optimal value R_1 is determined in Eq. 6.14. From $E(i) = E(1)$, we have

$$\frac{w_i^\alpha + c}{R_i^2 - R_{i-1}^2}(R^2 - R_{i-1}^2) = \frac{R_1^\alpha + c}{R_1^2} R^2.$$

Through simple manipulation, the above equation can be written as

$$\frac{w_i^\alpha + c}{w_i(w_i + 2R_{i-1})} = \frac{R_1^\alpha + c}{R_1^2} \frac{R^2}{R^2 - R_{i-1}^2} = a_{i-1}. \tag{6.15}$$

We obtain the following equation

$$w_i^\alpha - a_{i-1} w_i^2 - 2a_{i-1} R_{i-1} w_i + c = 0. \tag{6.16}$$

Notice that a_{i-1} depends solely on R_{i-1}. Thus, once R_{i-1} is known, we can compute w_i by Eq. 6.16. As R_{i-1} can be determined immediately from $R_{i-1} = w_{i-1} + R_{i-2}$, it turns out that w_i can be computed iteratively. That is, we compute w_2 first, and then w_3, and afterwards w_4, and so on. The resulting w_i is a function of R_1. We also have $\sum_{1 \le i \le q} w_i = R$. Hence, the value of q is also determined during the iteration when total width R is reached.

Balanced energy usage ($E(1) = E(2) = \cdots = E(q)$) is not achievable for $\alpha = 2$, regardless of values R, r_c, and c. Details about the derivation of this negative result can be found in Olariu and Stojmenovic (2006).

Note that energy balancing with adjusted transmission radii here assumed that each hop has the length equal to corresponding annuli width w_i. Such routing corresponds to routing along a straight line with sensors being available at desired locations. Naturally, high density of sensors are necessary to make use of this assumption, but even that may not be sufficient for energy balancing. Olariu and Stojmenovic (2006) were unable to actually design a data gathering scheme that will reasonably balance energy based on theoretical findings. Therefore this remains an open problem.

6.3 ENERGY EFFICIENCY BY SINK MOBILITY

This section briefly discusses how to achieve energy efficiency by exploiting sink mobility. Sink mobility may be classified as *uncontrollable* or *controllable*, in general. The former is obtained by attaching a sink node on a certain mobile entity such as an animal or a shuttle bus, which already exists in the deployment environment and is out of control of the network. The latter is achieved by intentionally adding a mobile entity for example, a mobile robot or an unmanned aerial vehicle, into the network to carry the sink node. In this case, the mobile entity is an integral part of the network itself and thus can be fully controlled.

6.3.1 Delay-Tolerant Scenarios

In a delay-tolerant WSN, for applications such as habitat monitoring and water quality monitoring, energy usage optimization embraces a lot of options. To maximize energy savings for sensors, direct contact data collection is the best option. That is, sinks visit (possibly at slow speed) all data sources and obtain data directly from them (Shah *et al.*, 2003; Gu *et al.*, 2005; Nesamony *et al.*, 2007; Sugihara and Gupta, 2008). This method completely eliminates the message relay overhead of sensors, and thus optimizes their energy savings. However, it has a large data collection latency for slow moving sinks. To reduce time delay, sinks may visit only a few selected rendezvous points (RPs) (Kansal *et al.*, 2004; Xing *et al.*, 2008, 2007), where sensor readings of all data sources are buffered and possibly aggregated, avoiding long travel distances at energy cost of multihop data communication. Both direct-contact data collection and rendezvous-based data collection can be supported by uncontrollable or controllable sink mobility.

Figure 6.2a depicts a taxonomy of existing approaches for energy-efficient data collection by mobile sinks in delay-tolerant WSNs. At the top level of the taxonomy are the two classes of collection methods, that is, direct-contact and

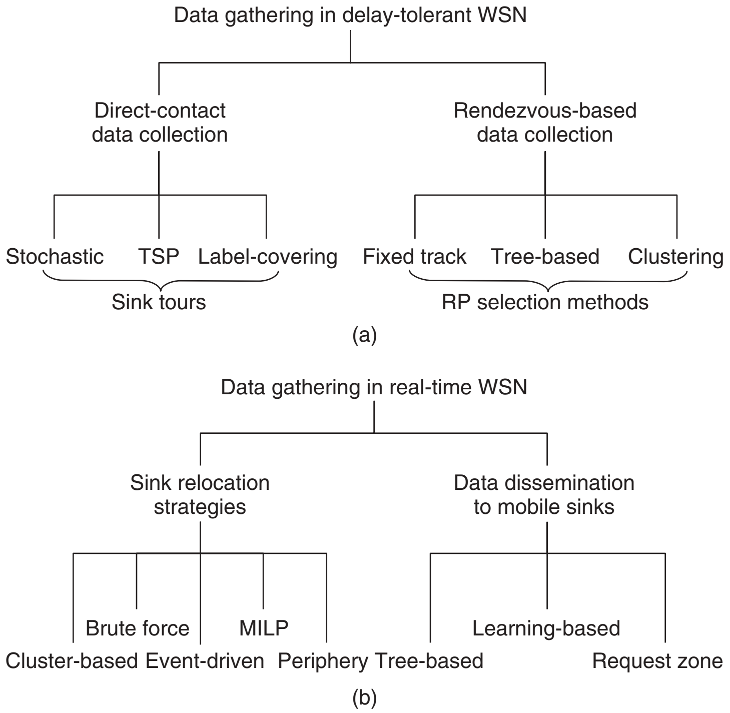

Data gathering in delay-tolerant WSN
Direct-contact data collection
Rendezvous-based data collection
Stochastic
TSP
Label-covering
Fixed track
Tree-based
Clustering
Sink tours
RP selection methods
(a)
Data gathering in real-time WSN
Sink relocation strategies
Data dissemination to mobile sinks
Brute force
MILP
Learning-based
Cluster-based
Event-driven
Periphery
Tree-based
Request zone
(b)

Figure 6.2 A taxonomy of energy-efficient data gathering by mobile sinks. (a) Delay-tolerant WSN. (b) Real-time WSN.

rendezvous-based. Each is further divided into three subclasses according to their employed techniques.

6.3.2 Real-Time Scenarios

In real-time WSNs, for applications like battlefield surveillance and forest fire detection, sensor readings ought to be collected in a timely manner by sinks. With effective mobile sink-based data dissemination (i.e., source-to-sink routing) methods, network lifetime can be prolonged by adaptively relocating sink nodes to positions with the largest energy gain as the network evolves. For example, Banerjee *et al.* (2009) suggested that sinks move toward data sources, or energy-intense areas, or the combination thereof; Luo and Hubaux (2005) concluded that the optimal sink mobility strategy is to move along the periphery of the network when the network has a circular shape and shortest path routing is used. Intelligent sink relocation requires controllable sink mobility. Uncontrollable (e.g., random or fixed track) sink movement may also balance energy consumption since the role of the "hot spot" rotates among sensors. But, it has a relatively inferior performance (Basagni *et al.*, 2008).

Figure 6.2b shows a taxonomy of existing approaches for energy-efficient data gathering in real-time WSNs. At the top level of the taxonomy are the two research subproblems, that is, sink relocation and data dissemination, each followed by representative solutions at the lowest level.

6.4 SINK MOBILITY IN DELAY-TOLERANT NETWORKS

In this section, we review the literature on energy-efficient data collection by mobile sinks in delay-tolerant WSNs. We examine direct-contact data collection methods first and study rendezvous-based data collection methods afterwards.

6.4.1 Direct-Contact Data Collection

In direct-contact data collection, a mobile sink collects data directly from data sources by one-hop communication. Sinks may retransmit data or, if needed, physically carry the data to a fixed base station. This approach minimizes energy consumption among sensors for communication, since sensors do not need to forward messages for each other. In this scenario, the main concern is the computation of the best sink trajectory that covers all data sources and minimizes data collection delay.

Stochastic Data Collection Trajectory

Shah *et al.* (2003) considered stochastic sink mobility and proposed a simple data collection algorithm. In their proposal, sensors buffered their measurements locally and wait for the arrival of a mobile sink. Multisink scenario is also considered. Each sink moves randomly and collects data from encountered sensors in its communication range. Collected data are then carried by the sink to a wireless access point (e.g., a fixed base station).

In the case of stochastic sink mobility, energy consumption at sensor side is only due to sink discovery and subsequent data transfer. Assume each sink broadcasts a beacon message while moving. A straightforward way of sink discovery is to monitor the wireless communication channel. Whenever a sensor hears the beacon message it concludes that a sink arrives. However, constant channel monitoring is very expensive in energy. Chakrabarti *et al.* (2003) show that, if sinks (e.g., mounted on shuttle buses) move along the regular path, then sensors can predict their arrival after being allowed a learning curve for their movement pattern.

After discovering a sink, data transfer should also start in an intelligent way. If a sensor simply transmits as soon as it discovers the sink, data may not be successfully delivered or may be delivered with many retrials, wasting energy. According to Anastasi *et al.* (2007a), message loss probability drops with decreased sensor-sink distance. Suppose the sink passes by sensors along straight line. To minimize energy consumption, data transfer should take place in the time interval with minimum message loss probability, which is exactly around the minimum sensor-sink distance point. From this consideration, Anastasi *et al.* (2007b) proposed an adaptive data transfer protocol. In Anastasi *et al.* (2007b), the contact time $\hat{f}(n+1)$ for the $(n+1)$th passage is estimated by the function $\hat{f}(n+1) = \alpha f(n) + (1-\alpha)\hat{f}(n)$, where $f(n)$ and α $(0 < \alpha < 1)$ represent the time elapsed since the previous (the nth passage) contact, the duration of contact, or the time between contact and data transfer, or other relevant measure (different measure has different function and its parameter). According to the estimation, sensors start data transfer properly in time and transmit a predefined number of bits. If contact time is large enough for sensors to perform a sleep–wake-up cycle before transmitting, they will do so to save energy.

TSP Tour for Data Collection

When sink mobility is a controllable factor, we can reduce data collection delay by properly selecting sink trajectory. It is not difficult to conclude that direct-contact data collection is generally equivalent to the NP-complete traveling salesman problem (TSP) (Lawler *et al.*, 1985). Informally, the TSP problem is: given a number of cities (i.e., sensors), find the shortest tour that visits each city (sensor) exactly once and returns to the starting city.

Nesamony *et al.* (2006, 2007) formulated the sink traveling problem as a variant of TSP, known as *traveling salesman with neighborhood* (TSPN), where a sink needs to visit the neighborhood of each sensor exactly once. The intuition is that it is sufficient for the sink to be within the communication range (modeled as disk) of a sensor in order to retrieve data from that sensor. Figure 6.3a comparatively shows the TSP tour (dashed thick lines) and the TSPN tour (thick lines) of four sensors for a mobile sink.

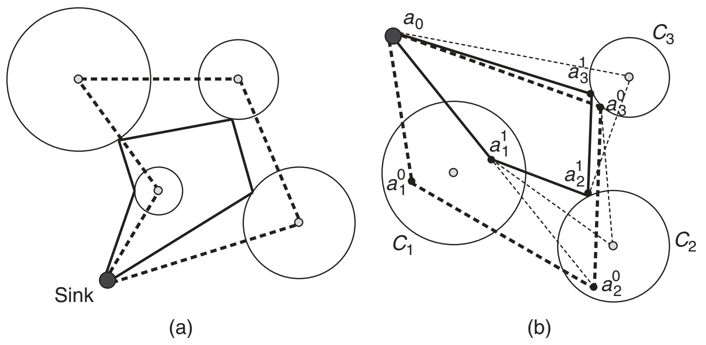


Figure 6.3 TSPN. (a) TSP with neighborhood. (b) Point set computation.

In Nesamony *et al.* (2007), the authors presented an algorithm for finding the best possible sink tour. This algorithm requires that the locations of all sensors are known. It first determines the visiting order of the disks. In this process, some ordering constraints may apply. For instance, the disks whose corresponding sensors are about to deplete their battery power have to be visited first in order to prevent data loss. If there are no constraints, then the most intuitive way is to order the disks based on the TSP order of their representative points. The representative point of a disk could be selected in different ways. For example, it could be a random point, the center point, or the closest point on the circumference to the starting point.

Once the visiting order is determined, the algorithm computes the optimal set of points accordingly. The initial set is composed of the starting point a_0 and the representative points a_i^0 of the ith disk, C_i. Then a_1^0 is updated to a_1^1 with respect to a_0, a_2^0, and disk C_1 as follows: if line $a_0a_2^0$ intersects C_1 then a_1^1 is any point between intersections; otherwise, a_1^1 is a point on the circumference of C_1 such that $|a_0a_1^1| + |a_1^1a_2^0|$ is minimized. In the latter case, the search space is reduced from the entire circumference of C_1 to the arc between the two lines from a_0 and a_2^0 to the center of C_1, and a binary search is used to find a_1^1. After a_1^1 is computed, a_2^0 is updated to a_2^1 with respect to a_1^1 and a_3^0, and so on. Finally, a_n^0 is updated to a_n^1 with respect to a_{n-1}^1 and a_0 and C_n. The sink tour defined by the new point set will have smaller length than the old one. The iterative update is repeated with the new point set as input until the length of the tour stabilizes. Figure 6.3b illustrates this process.

Sensors have limited storage space. They can only buffer a finite amount of data. Assume sensors have different data generation rate λ. Some sensors need to be visited more frequently (with respect to their buffer overflow time $o = \frac{b}{\lambda}$ where b is buffer size) than others so as to avoid data loss. Gu *et al.* (2005) addressed the impact of buffer limitation on the TSP for sink mobility and presented a partitioning-based scheduling (PBS) algorithm for sink mobility. In PBS, the locations of all sensors are known *a priori*. Sensors are partitioned into groups, called *bins*, such that sensors in the same bin B_i have their buffer overflow times in the same range, and the range of overflow times for B_{i+1} is twice that of B_i. Each bin is further partitioned into sub-bins according to sensor locations such that the sensors in the same sub-bin are geographically close to each other. This partition is realized by the k-dimensional tree (KD-tree) algorithm (Bentley, 1975).

The sink starts from the sensor with minimum buffer overflow time in a sub-bin of B_1. It travels along a so-called supercycle composed of a number of visit cycles of the bins. Each visit cycle contains exactly one sub-bin from each bin B_i in order. In each visit cycle, a sub-bin in B_i is followed by a geographically closest sub-bin in B_{i+1}. Because there are twice more sub-bins in B_{i+1} than in B_i, each sub-bin in B_i is followed by exactly two sub-bins from B_{i+1} in the supercycle. Figure 6.4 shows a supercycle of 4 visit cycles, where B_i^j is a sub-bin of B_i and each B_i^j contains only one node.

The sink traveling problem is reduced to the TSP in each sub-bin. Prim's algorithm (Prim, 1957) is used to compute the minimum spanning tree (MST) of sub-bins, and the order of visits is then determined by a preorder tree walk. Note that after the last sub-bin is visited, the sink moves to the closest sensor in the next sub-bin instead of returning to the first visited sensor in current sub-bin. Once the path is constructed, the minimum sink speed for lossless data collection can be determined by $\frac{L_{\max}}{o_{\min}}$, where $L_{\max}$ is the length of longest path between two consecutive visits to a sensor, and $o_{\min}$ is the minimum buffer overflow time.

Gu *et al.* (2006) studied sink mobility scheduling for the differentiated message delivery problem, where periodically generated regular messages are delivered without sensor buffer overflow and aperiodically generated urgent messages delivered within a deadline Δ. The PBS algorithm (Gu *et al.*, 2005) produces a schedule, where the intervisit duration of a sink to every sensor n_i is not larger than the effective overflow time $eot(o_i)$ associated with the sensor's buffer overflow time o_i, that is, the minimum overflow time of the bin where n_i resides. However, if $eot(o_i) > \Delta$, the PBS solution does not guarantee urgent messages to be delivered in time. Gu *et al.* (2006) suggested to deliberately reduce the *eot* of some sensors and allow multihop message relay to handle this situation. It is realized by a new algorithm, multihop route to mobile element (MRME) with PBS (Gu *et al.*, 2005) as a subroutine.

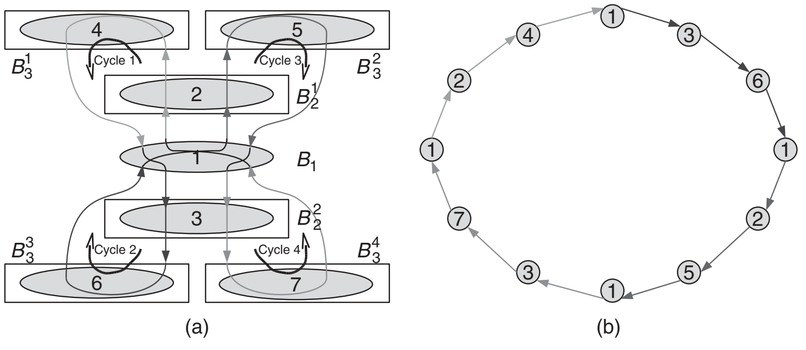


Figure 6.4 A supercycle composed of four visit cycles. (a) View of bins. (b) View of nodes.

Urgent message delivery deadline can be satisfied at covered sensors, that is, sensors where $eot \leq \Delta$. In MRME, urgent messages generated at uncovered sensors do not have to wait for on-site pickup; they are relayed to nearby sensors within $d_{\max}$ (a predetermined value) hops (from their originators) that are visited more frequently by the sink. Let t_{tr} be the transmission delay per hop. If an urgent message generated at sensor n_j is sent to a d-hop ($d \leq d_{\max}$) neighbor n_i, then for lossless scheduling, the intervisit duration of the sink to n_i should be at most $eot(o_i) \leq \Delta - d \times t_{\text{tr}}$. For n_i to cover its n_j, it should be visited by the sink at least at frequency $\frac{1}{\Delta - d \times t_{\text{tr}}}$.

For sensor n_j, buffer overflow time reduction will cause increase of its sink-visit frequency $f_j = \frac{1}{eot(o_j)}$. Define the relative increase as $F_j = \frac{f_{\text{new}}(j) - f_{\text{old}}(j)}{f_{\text{old}}(j)} = \frac{eot_{\text{old}}(o_j) - eot_{\text{new}}(o_j)}{eot_{\text{new}}(o_j)}$. Denote by $C(n_i, d)$ the set of uncovered d-hop neighbors of n_i. The gain at n_i due to overflow time reduction within its d-hop neighborhood $N_{i,d}$ is defined as $Gain(i, d) = \sum_{n_j \in C(n_i, d)} F_j - \beta \times F_i$, where β is a system parameter used to adjust the behavior of the algorithm. Further define the worst case delay D_i for urgent messages generated at node n_j as $D_i = \min_{d \leq d_{\max}} \{d \times t_{tr} + \min_{j \in N_{i,d}} \{eot(o_j)\}\}$. With the above notations, the skeleton of algorithm MRME is described below.

The execution of algorithm MRME has three phases. In the first phase, sensors are partitioned into sub-bins as in PBS, and uncovered nodes n_i are identified by checking the satisfaction of inequality $D_i > \Delta$; in the second phase the buffer overflow times of some nodes are iteratively reduced until no uncovered sensor exists; in the third phase, PBS is run with modified overflow times to produce a sink mobility schedule. In each iteration of the second phase, the maximum $Gain(i, d_0)$ for $n \leq d_{\max}$ is found, and the buffer overflow time of n_i is reduced to $\Delta - d_0 \times t_{tr}$; then the uncovered node set is recomputed, and the minimum overflow time $o_{\min}$ in the network is updated.

Label-Covering Tour for Data Collection

Sugihara and Gupta (2007, 2008) addressed sink path selection for data collection delay minimization. They waived the requirement for exact one-time visit of the sink to each sensor's neighborhood. The intuition is that the sink's travel time could be long if the length of the intersection of the its path and the communication range of each sensor is short, because, in that case, the sink has to slow down to collect all the data. Exact one-time visit may not always be a winning strategy. Multivisits together with proper speed control may yield a better solution.

The authors simplified the path selection problem by reducing search space to a complete geographic graph, where there are vertices at sensors' locations and the sink's initial location. The sink is assumed to move in this graph along edges from vertex to vertex. Each edge is associated with a cost and a set of labels. Cost is defined as Euclidean length of the edge; the label set represents the set of sensors whose communication ranges intersect with this edge, that is, the sensors that the sink can collect data from while traveling along this edge. Figure 6.5 shows such a complete graph constructed over a network of six sensors. In this

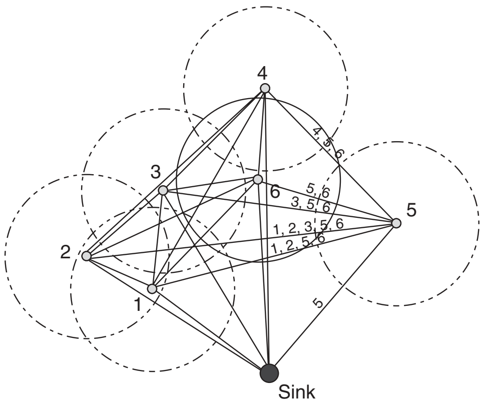


Figure 6.5 Complete graph of sensors and the sink node.

figure, sensor communication ranges are marked by dashed circles, and label sets associated with the links incident to node 5 are displayed.

The objective is to find a minimum-cost tour along which the sink can collect data from all the nodes. In other words, a shortest tour whose associated label set covers all sensors. In this setting, the sink does not necessarily visit all vertices. The authors proved that the shortest label-covering tour problem is NP-hard, and presented an approximation algorithm to solve it. The algorithm first finds a TSP tour T by any TSP solver. Then, by dynamic programming, it finds the shortest label-covering tour that can be obtained by applying shortcutting to T. Using the speed control algorithms and the job scheduling algorithm presented in Sugihara and Gupta (2007), the authors experimentally validated the effectiveness of the algorithm and showed that it has better performance than TSP-like algorithms when sensors have large communication ranges.

6.4.2 Rendezvous-Based Data Collection

Direct-contact data collection has great advantage for energy savings. However, it significantly increases data collection latency because of the sinks' low moving speed. Rendezvous-based data collection is proposed to achieve trade-off of energy consumption and time delay. Sensors send their measurement to a subset of sensors called *rendezvous points* (RPs) by multihop communication; a sink moves around in the network and retrieves data from encountered RPs. The use of RPs enables the sink to collect a large volume of data at a time without traveling a long distance and thus greatly decreases data collection delay. Relevant research focuses mainly on RP selection. Note that, since RPs are static, data dissemination to RPs is equivalent to data dissemination to static sinks, which has been intensively studied in traditional static WSN.

RP Selection by Fixed Track

Kansal *et al.* (2004) proposed to use a straight line sink path for data collection. At an initialization phase, the sink broadcasts a beacon message while moving along a straight line. The message has a hop count field indicating the number of hops it has traveled. Every receiver node rebroadcasts the message if and only if the message has a smaller hop count than that in memory. It increments the hop count field before rebroadcasting. It also remembers the node from which it receives the message. After the initialization phase, a number of trees are constructed, each rooted at a node along the sink path, and each node belongs to exactly one such tree.

The root of every tree is taken as an RP. Sensors subsequently send their measurement along the upward path to the root of their residing tree. As the sink moves, RPs send their own data together with the data received from their tree members to the sink. Two motion control algorithms were presented to adjust sink speed to increase the amount of collected data. In the stop to collect data (SCD) algorithm, the sink stops for a while at locations where sensors are found waiting with data. In the other algorithm, the sink moves slower in regions where the data delivery success rate is moderately poor and temporarily stops in regions where data loss is severe.

Multisink scenarios were considered in Jea *et al.* (2005). The sensory field is divided into equal-sized areas, each having a sink. Then, the single-sink algorithm is run in each area. Randomized sensor distribution may cause unbalanced load (i.e., sensor assignment) among sink paths. A load balancing algorithm is presented to ensure that each sink path is assigned the same number of sensors. This algorithm is executed by an elected sink under the assumption that sinks can always communicate with each other and thus can exchange their sensor assignment information.

Xing *et al.* (2008) considered the case that the sink is allowed to move only along a fixed track. They assumed that sensors have the same transmission range and are densely deployed. In such a network, the total energy consumption for message transmission along a multihop path is proportional to the Euclidean distance between sender and receiver. Further, data aggregation is applied at each sensor node. The objective is to select RPs along the sink track such that the total length of edges that connect sources to RPs is minimized.

An MST-based algorithm rendezvous design for fixed tracks (RD-FT) was presented. In this algorithm, an *optimal* set MST_{sT} of MSTs that connect all sources to the sink track (sT), in the Euclidean domain. Each individual MST in the set does not necessarily span all data sources. The set is optimal in that the length sum of its member MSTs is minimal. Each MST in MST_{sT} satisfies the following two conditions: (i) it is rooted either at the sink starting point, an end point, or a turning point of, or at the projection point of a data source on, sT; (ii) for any of its contained data sources, the length of the tree path to the root is smaller than the distance to any other point on sT. Figure 6.6 shows seven data sources and an sT (the zigzag line) between points X and Y. In this example, MST_{ζ} contains 5 MSTs, respectively rooted at points A, B, C, D, E on the sT. Note that node 6 is linked to node 7 rather than to the closest point E on the sT because link 6–7 is shorter than link $6-E$ (and therefore this local MST is shorter that way).

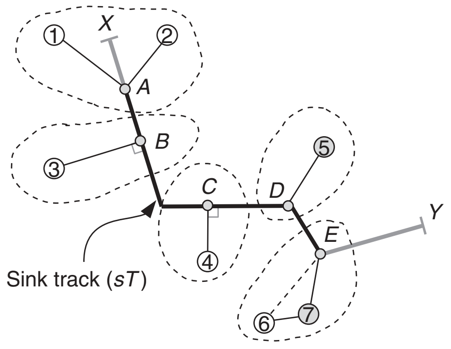


Figure 6.6 Rendezvous design for fixed tracks.

Set MST_{sT} are approximations of the optimal reporting trees in practice for data gathering. Thus algorithm RD-FT takes the roots of these trees as RPs. It adopts the Kruskal's algorithm (Kruskal, 1956) (with minor modifications) to find MST_{sT}. After MST_{sT} is constructed, the RPs are found. Then the actuator (sink) tour can be reduced only to the portion of the track that covers these points. For example, in Figure 6.6, the sink will travel only between A and E.

RP Selection by Reporting Tree

Xing *et al.* (2007, 2008) studied reporting tree-based RP selection subject to the data collection deadline D. Rendezvous points must be properly selected from a data reporting tree such that the sink tour of the RPs is not longer than the maximum distance L that the sink can travel within the time D.

In Xing *et al.* (2007), the authors considered a predefined reporting tree rooted at a static base station BS. In this tree, nodes shared by multiple data reporting paths are called *junction nodes*. Suppose that the locations of source nodes and junction nodes are known and that nodes are densely deployed. Then the reporting tree can be approximated by a geometric tree TR rooted at BS and composed of source nodes and junction nodes. Any point on an edge of TR can serve as RP. Both constrained and unconstrained sink mobility are studied. A greedy algorithm rendezvous planning with constrained path (RP-CP) was presented for sink mobility constrained on TR. Each edge of TR is assigned a weight, equal to the number of sources in the subtree rooted at its upper end (the end toward the root). Sort the tree edges in the decreasing order of their weights. Rendezvous planning with constrained path greedily adds edges of maximum weight to an edge set W (which is initially empty), without creating cycles, such that the total edge length of W is not larger than $L/2$. Part of the next unchosen edge may be included in W to ensure its edge length sum is exactly $L/2$. The final W is a connected subtree, and the nodes in W are RPs. The sink traverses W in preorder, resulting in a tour of length exactly L. It is proven that the preoder walk of W is an optimal tour when sink path is constrained on TR.

A greedy heuristic algorithm RP-UG (rendezvous planning with utility-based greedy heuristic) was presented for free sink mobility. RP-UG adds virtual nodes to TR such that every tree edge is not longer than a predefined value L_0. It operates in iterations. In each iteration, a node in TR with the greatest utility is included in a RP list (which, initially contains only BS). The *utility* of an RP is defined as the ratio of the network energy saved by adding it on the sink tour to the length increase of the tour. The length of the tour of RPs is computed using a TSP algorithm. The addition will cause utility change of the RPs in the list. All the RPs whose utilities become zero are immediately removed from the list. If the maximum tour length is reached, or if all source nodes are included in the list, RP-UG terminates; otherwise, a new iteration is started to find more RPs. By adjusting L_0, one can achieve desirable trade-off between solution quality and computational complexity.

A Steiner minimum tree (SMT) (Hwang *et al.*, 1992) is a tree of shortest length connecting a given set of points. It differs from a MST in that it may contain extra intermediate points, called *Steiner points*, in order to reduce the length of the tree. For the Euclidean Steiner problem, each Steiner point must have three incident edges of mutual angle $2\pi/3$. Figure 6.7 shows a SMT. In this tree, circular nodes are given points, and square nodes are Steiner points. Generally speaking, SMT construction is an NP-complete problem, and heuristics are used in practice. As SMT has minimum total length, it leads to optimal total energy consumption for data dissemination to sinks when used as reporting tree, and it also serves as lower bound of the optimal TSP tour of data sources.

From the above consideration, the authors presented a SMT-based algorithm rendezvous design for variable tracks (RD-VT) in Xing *et al.* (2008), under the assumptions of dense node distribution, known source locations, and free sink mobility. The rational of RD-VT is to use SMT as a reporting tree and restrict RPs to nodes of the SMT, such that the RPs form a subtree and they can be visited by a sink tour no longer than L while the total edge length of the rest of the SMT is minimized. In RD-VT, an SMT of sources is constructed with an arbitrary source as root. It is walked in preorder up to distance $L/2$, and visited tree nodes are taken as RPs. Tree walk is recursively extended on the next tree edge by a length of half of the difference between the TSP tour length and L, until the TSP tour of selected RPs is sufficiently close to L. The preorder tree

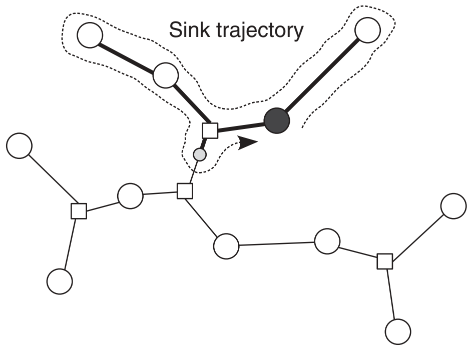


Figure 6.7 Steiner minimum tree (SMT).

walk on the final subtree defines sink tour; sources send data along the SMT to RPs. Figure 6.7 shows that the sink tour computed by RD-VT. In this figure, white circles represent sources; the black circle is the root of the SMT.

RP Selection by Clustering

Rao and Biswas (2008) presented a generic framework for mobile sink-based data gathering by integrating several existing algorithms. In this framework, a minimum k-hop dominating set is constructed. All nodes in the dominating set are called *navigation agents* (NAs). Two adjacent NAs are at least $k + 1$ and at most $2k + 1$ hops away from each other. By tuning the parameter k, the framework migrates from direct-contact data collection, through rendezvous-based data collection, to traditional static sink-based data gathering.

Each NA constructs a minimum hop tree rooted at itself and spanning up to a depth of $2k + 1$ hops. During tree construction, it identifies adjacent NAs and meanwhile constructs shortest paths to them. The nodes along such a shortest path are called *intermediate navigators* (INs). They are used to navigate the mobile sink to move between RPs. The NA-rooted trees will be used for subsequent data dissemination. Rendezvous points and INs together constitute a connected overlay graph. A distributed ant colony optimization-TSP algorithm (Bonabeau *et al.*, 1999) is adopted to find a TSP tour of NAs for the sink over the overlay graph. After the TSP tour is found, NAs know their next NAs in the tour.

Every NA transmits a hello message periodically. The sink starts to move from an arbitrary location and attempts to discover a local NA by listening to the hello message. Once the first NA is discovered, it moves toward the NA according to the received signal's direction of arrival (DOA). Afterwards, it starts to travel along the TSP tour of NAs bypassing INs similarly by following the DOA of those nodes' signals. The use of DOA enables the scheme to work in the absence of any location information.

The one-hop neighbors of a NA are called *designated gateways* (DGs). Sources that are not adjacent to the sink tour send their data toward the root using their residing NA-rooted tree constructed during IN node identification process. Data stops at the closest DG on its way toward the root, and is buffered at that DG. In this sense, DGs are actually RPs. The benefit of buffering data at a DG rather than at a NA is twofold. It saves message transmission and thus energy consumption by reducing data communication path by one hop; it avoids the huge aggregated storage load at NAs. Along its TSP tour, the sink retrieves data from encounter NAs and their DGs.

From the above description, we can see that by adjusting the value of k, a desired balance of data collection delay and energy expenditure can be achieved. When $k = 1$, the sink visits every node's communication range, and thus the framework turns into a direct-contact data collection scheme. When $k = k_{\max}$ (an obvious value for $k_{\max}$ is network size n), there is only one NA in the network, which is dominating all the other network nodes. In this case, once the sink reaches the only NA, it stops moving, resulting in static sink scenario.

6.5 SINK MOBILITY IN REAL-TIME NETWORKS

In this section, we study sink mobility for energy efficiency in real-time networks. We consider controllable sink mobility only. We first introduce representative sink mobility strategies and then review some specialized routing algorithms for data dissemination to mobile sinks.

6.5.1 Sink Relocation

According to the energy models introduced in Section 6.2, a sink should move toward data sources so as to shorten path length and thus reduce and balance energy consumption. During relocation, the sink may keep receiving data. It will bring extra load to the nodes in its visited areas and increase their energy consumption. So it is desirable that the sink goes through energy-intense area rather than energy-sparse areas. On the basis of this consideration, Bi *et al*. (2007) suggested that the sink temporarily changes its moving direction by a certain policy before entering an energy-sparse area and later turns back toward the relocation destination.

Optimal multisink placement is a NP-complete problem, as it is equivalent to the NP-complete dominating set problem on unit disk graphs (Bogdanov *et al*., 2004). In the following sections we will introduce some representative sink relocation strategies presented in the literature.

Cluster-Based Approach

Banerjee *et al*. (2009) studied sink relocation in a WSN clustered with multiple mobile sinks. They assumed that each sensor joins one and only one nearest cluster and sends data to the corresponding sink (clusterhead). Sinks form a connected overlay network for further data fusion to a stationary base station. Sinks are assumed to be initially well dispersed such that each cluster covers roughly the same area of the network without large overlapping, and that the entire network is fully covered. Clusters remain unchanged once constructed.

Sinks move only within their own clusters. While moving, they repeat route discovery in the overlay network so as to preserve a route to the base station. If no route can be discovered, they move back to their previous location. This connectivity preservation method can be enhanced by making use of one- or two-hop neighborhood information. The authors analyzed energy consumption in the network when sinks move independently and randomly by slightly modifying the energy model introduced in Section 6.2. They presented three controlled sink mobility strategies.

In a residual energy-based strategy, a sink always moves toward the residual energy center of its cluster to balance energy consumption. The residual energy center is the average of the positions of cluster members (sensors) weighted by their residual energy. Each cluster member sends its remaining energy level and locally estimated (based on history) energy dissipation to the sink, which then predicts the sensor's energy accordingly if necessary. The residual energy center could however be far away from the current location of events, causing increased distance of data transmission, and thus increased total energy consumption.

In an event-based strategy, a sink always moves toward the event region, that is, the region with maximum data flow, in its cluster. The objective is to shorten data transmission path, reduce sensors' overhead of relaying messages, and eventually increase network lifetime. However, after the sink reaches the event center, it will stop there. In this case, the set of relaying nodes will remain unchanged and further energy balancing is prevented.

Because of limitations of these two strategies, the authors further suggested to combine them to obtain a hybrid strategy. A sink first moves toward the event center then toward the energy center. When a sink moves away from an event center, while another moves close to it, the sensors sensing the event will report to the sink that yields larger energy gain.

Event-Driven Approach

Vincze *et al*. (2007) addressed single-sink relocation problem in an event-driven WSN, where sensors have the same sensing radius and become data sources only when they detect events. Data sources send their readings to the sink through message relay using shortest paths; other sensors do not transmit. An event is modeled as intruder moving at a fixed speed and in a random direction. It is assumed that the sensory field is a circular region, and that sensors are distributed uniformly and densely enough such that each source-to-sink path approximates a straight line and consists of hops of equal length.

Through approximation, the authors show that the average transit load (the load of forwarding messages) L_P of a node P for an event is proportional to the distance between P and the event and reversely proportional to the distance between P and the sink. They also show that L_P is maximized when P is only half a hop away from the sink. Then, from the sink's view, the most heavily loaded sensor is in the direction toward the farthest event.

On the basis of the above results, they concluded that, to achieve lowest total energy consumption, the sum of event distances needs to be minimized by relocating the sink. This is equivalent to the well-known "facility location" problem. They further indicated that, to have a balanced energy consumption among nodes, the maximum sensor load should be minimized. That is, to minimize maximum event distance from the sink. This is equivalent to the "minimal enclosing circle" problem in the case of straight line routing.

The authors suggested that the sink may predict the future location of current events based on historical event data and make relocation decisions in advance. Once the target position is computed, the sink moves there. As events are changing their positions over time, the sink will keep adjusting its location according to the evolution of current events.

Brute Force Approach

Friedmann and Boukhatem (2009) presented a centralized brute force algorithm for multisink relocation. The algorithm is run periodically to check if sinks should be relocated. Sink relocation takes place if and only if the new sink positions leads to a reduced total energy cost. Sinks are assumed to have a global view of the network, and thus are able to run a centralized algorithm.

Each edge in the underlaying network graph is assigned a weight for each of the two transmission directions. The weight is subject to the sum of two factors, the inverse of remaining energy of the receiver node and the energy consumed by message transmission along the edge, balanced with their respective coefficients. As node remaining energy keeps decreasing, edge weight changes over time. A centralized source routing approach (e.g., Dijkstra algorithm) is used to find shortest path (i.e., the path with minimum weighted sum) from each sensor to its nearest sink. Such a path attempts to avoid using energy-scarce nodes and reduces total energy consumption along the path for message transmission.

In order to minimize total energy consumption in the network, sinks should be placed at positions that yield smallest total energy cost, which is measured by the weighted sum of all the shortest paths. When computing total energy cost, the weight of paths linking active nodes (i.e., nodes that are sensing and transmitting) to sinks are intentionally increased, so as to make sinks close to active nodes and reduce energy consumption for message relaying.

A heuristic approach is used to find best sink positions. In this approach, relocation is restricted to only a few (1, 2, or 3) sinks in each round, and optimal solutions are found in those cases. To further reduce search space, sinks are allowed to move only in eight geographic directions: north, south, east, west, northeast, northwest, southeast, southwest. The authors considered the transitional effect (during relocation, communication cost might temporarily increase). Instead of moving directly to optimal positions in one step, sinks apply multiple-step movement, where each step is determined by a constrained local search in a bounded local area.

MILP-Based Approach

Basagni *et al.* (2008) addressed single-sink relocation by modeling it as a mixed integer linear programming (MILP) problem. The sensory field is partitioned into a two-dimensional grid. The set S of grid points, called *sink sites*, are the candidate locations that the sink may visit. The sink moves step by step, each step to a site within a predefined range d_{MAX}; it has to stay at a site for at least a predefined $t_{\min}$ number of time units. When the sink is traveling, it is considered not reachable, and thus sensors hold their data and do not transmit. The holding time can be tuned by adjusting the granularity of grid division and the value of d_{MAX}. Network-wide flooding is used by the sink to notify sensors of its location and reachability.

Sink site set S and maximum step-moving distance d_{MAX} define a graph. The MILP problem is to determine the initial sink site, sink relocation path, and sink sojourn time t_k at each site k over this graph so that an objective function $\sum_{k \in S} t_k$ is maximized. This objective function describes the *effective network lifetime*, namely, the time period when sensors are able to transmit their reports. It is subject to a number of constraints. For example, the energy spent by a node for data delivery and routing cannot exceed its initial energy.

Centralized integer linear programming (ILP)-based solutions are not scalable. For larger networks, the cost of gathering network information at a single node, and the computational cost to run an ILP solution, rapidly increases, making such solutions feasible only for networks with few tens, and not few hundreds of nodes. In addition to the centralized MILP-based solution, the authors also presented two localized sink relocation strategies: greedy maximum residual energy (GMRE) strategy and random movement (RM) strategy. Sink moves to an adjacent site where sensors have maximum residual energy; the latter requires the sink to move to a randomly selected adjacent site.

Periphery Approach

According to Luo and Hubaux (2005), when a WSN has a circular shape, and when shortest path routing is used, the optimal sink mobility strategy is to move along the periphery of the network. Following this result, Hashish and Karmouch (2009) suggested to deploy sinks along the network outer boundary and presented a simple sink relocation scheme to deal with the sink partition problem caused by fast energy depletion of nodes around sinks.

In the proposed scheme, the outer boundary of the network is identified at an initial phase. Sinks divide the boundary into a number of segments. Figure 6.8 shows a circular-shaped network with four sinks $S_1, \ldots, S_4$, which partition the outer boundary (boundary nodes are highlighted) into four segments. In each segment, every boundary node maintains the hop count to the closest sinks in both directions. A sink is elected to compute the virtual center (v-center) of the network. Given a predefined parameter h, boundary nodes that are $k * h$ hops away for $k = 1, 2, \ldots$ from their nearest sink send their locations to it. The reports from all sinks are (in aggregated form) collected at the elected sink. The elected sink then computes the center of the virtual area defined by these nodes as v-center and floods the entire network with the location of v-center. In Figure 6.8, the computed v-center is near the center of the sensory field.

When a sensor detects an event, it sends event data away from the v-center toward the outer boundary using directional forwarding. After a boundary node receives sensor data, it forwards the data along network boundary to the closest sink. By this data dissemination method, two data sources a and b in Figure 6.8, send data to sinks S_1 and S_3, respectively. As sinks are located along the boundary, the boundary node will deplete their energy faster than internal nodes, leading to so-called *peeling phenomenon*. If a sink finds it is partitioned from the network, or having a low number of one-hop neighbors, or receiving data at low rate, it moves a distance of its transmission range toward the v-center to maintain connectivity. For example, in Figure 6.8, because the three neighbors of sink S_4 all run out of their power, S_4 decides to move to position P.

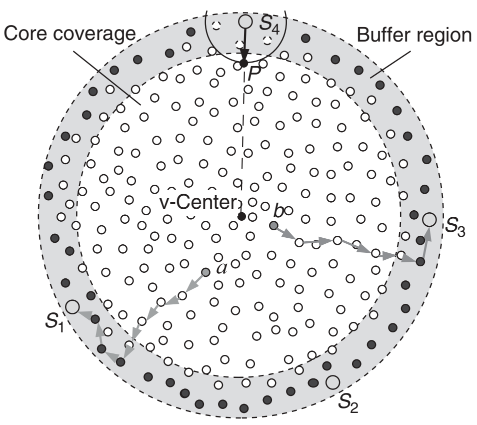


Figure 6.8 Sink deployment, source-to-sink routing, and sink relocation.

The undesired peeling phenomenon causes coverage decrease. To ensure coverage, the authors suggested that more sensors or a buffer region (the gray zone in Fig. 6.8) be used along the core coverage area. On the basis of the energy consumption model introduced in Section 6.2.2, the authors concluded that the proposed sink relocation scheme leads to a suboptimal energy balancing.

6.5.2 Data Dissemination

Data dissemination deals with sensor-to-sink data communication. Data flows from a data source to a sink are considered *downstream*; the reverse are called *upstream*. At the core of data dissemination is the problem of routing to sinks. Many routing protocols were developed for WSNs with static sinks. In the presence of sink mobility, data dissemination is a combined problem of location service (see Chapter 8) and routing (see Chapter 4), where nodes share a few common destinations, that is, sinks.

Recently, Wu and Chen (2007) addressed the offset problem of network-wide flooding-based sink location update with energy saving from sink mobility, and presented a dual sink scheme. In this scheme, a mobile sink updates its location by range-restricted flooding for energy saving; sensors that do not have the latest location of the mobile sink send data to an alternate known static sink. This scheme does not solve the actual problem of routing to mobile sinks. Specifically tailored nonflooding-based solutions exist in the literature. Below, we will review some representative related works on the topic.

Tree-Based Approach

Kim *et al.* (2003) presented a scalable energy-efficient asynchronous dissemination (SEAD) protocol. For each data source, SEAD constructs an SMT like the data dissemination tree, called *d-tree*, rooted at the source. Initially, a d-tree contains its root node only. Each sink chooses a nearest neighboring sensor as an *access node*. When it wants to receive data from a source, it joins the d-tree of that data source via its access node. In a d-tree, leaves are sink access nodes, and internal nodes are added Steiner nodes, called *replica nodes*. Figure 6.9 shows the d-tree of source B.

When a sink S_i wants to join the d-tree of source B, it directly (not through d-tree) sends B a join query via its access node A_i through the underlaying routing protocol. This query message contains the location of A_i and the desired data rate R_i of S_i. After B receives the message, it initiates an iterative search phase to find a *gate replica*, which is a replica node through which A_i will be connected to the d-tree. The search starts from B, and proceeds downward along the d-tree in accordance with the result of an energy saving test conducted at each visited tree node r. Denote by $K(r)$ the additional cost incurred by connecting A_i to node r. In the energy test, r calculates $F = K(r) - K(h)$ for every tree child h. If all the results are less than zero, the energy test is passed, and r becomes the gate replica; otherwise, it forwards the message to a child that maximizes F. If no such a satisfying node is found, the search will finally reach a leaf node. In this case, the leaf node's parent is taken as a gate replica. Figure 6.9 illustrates this search phase, which stops at node g.

We now examine how A_i is connected to the discovered gate replica g. There are two possible connection modes, as shown in Figure 6.9. In a *nonreplica* mode, A_i is connected as child to g. In a *junction replica* mode, a new replica node k is first created as child of g, and the tree link between g and one of its children c is removed, and A_i and c are then both connected to the new replica node k. Through local computation, gate replica g chooses the connection mode that yields smaller energy cost around itself on the d-tree. When the junction replica mode is to be used, the computation also determines the child c and the neighbor r that will together lead to the new replica. In this case, g sends r a message containing all necessary information for further computation. Upon receiving the message, node r repeats the above process with respect to c and its own neighbor set. The node k where the nonreplica mode is selected becomes the new replica node; then A_i is connected to k, and c becomes its sibling. If necessary, the gate replica g informs its parent to increase data rate to R_i.

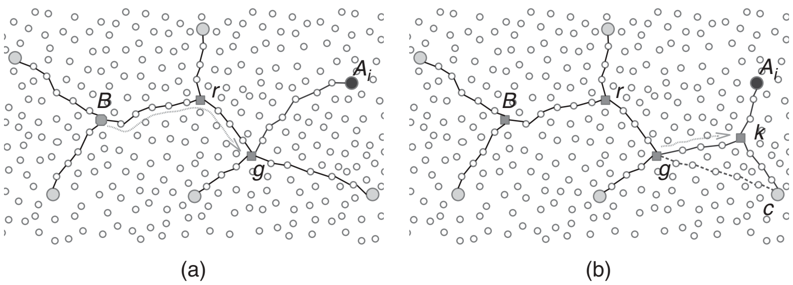


Figure 6.9 SEAD. (a) Nonreplica mode. (b) Junction Replica Mode.

As a sink S_i moves, if the hop count between S_i and its access node A_i exceeds a threshold value H_i, it replaces its access node with a new one A'_i, a currently closest neighbor. If $|A_i A'_i|$ is less than a threshold value T_m, $A_i{}'$ is simply connected to A_i without changing the structure of the d-tree; otherwise, S_i sends via A'_i its latest position to B, which then triggers the replica search phase to update the d-tree. We can see that the performance of this algorithm, SEAD depends very much on the selection of T_m. If T_m is too large, the optimality of the data dissemination path degrades; if T_m is too small, the maintenance cost of d-tree will be will be high. In the case of unpredictable or random sink mobility, it is difficult to chose a proper value for T_m in advance.

Learning-Enforced Approach

Baruah *et al.* (2004) presented a hybrid learning-enforced time domain routing (HLETDR) algorithm for data dissemination to a mobile sink. The sink moves following a certain fixed pattern. A *sink tour* is defined as the smallest time duration after which the sink's movement pattern repeats itself. Each node divides the sink tour evenly into a predefined number of time domains and assigns a weight to all its neighbors for each domain. The weight indicates at a certain time, what is the best way to forward a packet to the sink. Initially, the neighbors have equal weight, implying that they are equally good for data dissemination. The goal is to find the best routing path from each sensor to the sink in different time domains. It is accomplished by properly adjusting a sensor's neighbors' weight through a learning process, explained below.

The nodes whose vicinity is periodically visited by the sink are called *moles*. Figure 6.10 shows three moles along the sink path. Each data source spontaneously pushes data to the sink through multihop message relay. A data packet is forwarded by each intermediate node to the neighbors with the highest weight. Multiple copies of the packet may be transmitted simultaneously in the network. All these copies will finally reach, and are buffered at a mole. When the sink visits a mole, it obtains data from the mole. For example, in Figure 6.10, where the sink moves from north to south along the indicated track, three copies of a data packet are transmitted from data source S along three paths toward the sink at certain time domain, and they respectively reach moles X, Y, and Z.

Each mole learns the sink's movement pattern over time, and statistically characterize sink arrival time within a sink tour as a Gaussian distribution function. Thus it is able to estimate the likelihood that the sink is in its vicinity in any time domain, and evaluate the goodness G of a data transmission path when it receives the data at certain time. The value of G is high if data arrival time is close to the mean of sink arrival time, or low otherwise. For example, in Figure 6.10, the G values computed by the three moles X, Y, and Z are relatively low, high, and medium to each other. Note that if a mole does not have enough storage space to hold the expected amount of traffic, or the variance in the mobility pattern is very high, or certain time constraint for delivery needs to be satisfied, mole-to-mole data propagation along sink path is performed until a RP with a sink is reached.

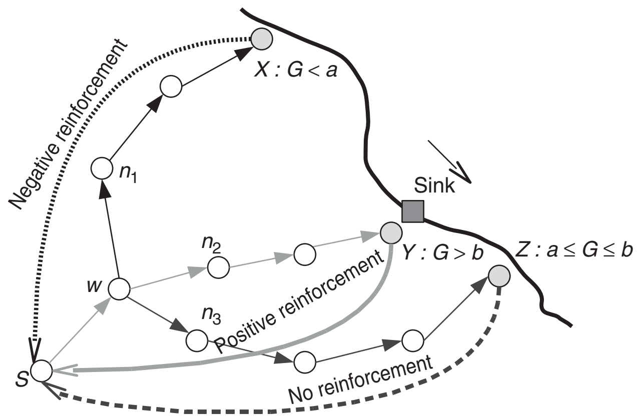


Figure 6.10 HLETDR.

After computing the goodness value G for a data transmission path, a mole sends value G to the data source along the reverse path. In the path, every intermediate node w updates the weight for its next hop (i.e., downstream neighbor toward the sink) n, based on value G and two predefined threshold values a and b. If $0 < G < a$, the weight of n is decreased, which is called *negative reinforcement*, meaning that m is not a good next hop to the sink in the current time domain. If $a \leq G \leq b$, n's weight remains unchanged, that is, *no reinforcement*. If $b < G < 1$, the weight of n is increased, which is referred to as *positive reinforcement*, meaning that n is on a good path to the sink in the current time domain. In Figure 6.10, moles X, Y, and Z respectively initiate negative reinforcement, positive reinforcement, and no reinforcement processes for the current time domain. By these means, node w is able to locally select best path with respect to time domains for timely data delivery in the future.

Request Zone Approach

Ammari and Das (2005) presented a Weighted Entropy DAta diSsemination (WEDAS) scheme. This scheme is a variant of request zone location service (refer to Chapter 8). It implicitly assumes that sensors have bounded adjustable transmission radii, and uses additional energy-aware next hop selection for routing to a mobile sink. The novelty stems from the quantification of the uncertainty of nodal remaining energy. Below, we will elaborate on this scheme and discuss a possible improvement.

Suppose the sink S is currently located at position a_1 and will relocate to position a_2. The relative mobility zone $Z_m(D)$ of S is defined as the circular area of diameter $D = |a_1a_2|$. Before leaving a_1, S advertises $Z_m(D)$ by flooding entire network with the two positions a_1 and a_2; during the course of its relocation, S does not do the advertisement any more.

Each sensor s_i computes $Z_m(D)$ from the received sink advertisement and identifies a subset $CNS(s_i)$ of neighbors, called coordinator node set. CNS consists of nodes that exist in the area (i.e., request zone in request zone location service) defined by its communication range and the two line segments s_ia_1 and s_ia_2. Nodes in this set are all neighboring s_i in the direction of the mobile sink. When sending messages to S, s_i routes the messages toward the center c of $Z_m(D)$ through a node form this set.

Sensor s_i maintains two vectors for every node s_j in $CNS(s_i)$. One contains the accurate remaining energy of s_j at k different time instants, which were advertised by s_j; the other includes the estimated remaining energy of s_j at those time instants. The *relative energetic distance* between the two vectors is the average of the absolute value of the distance between all corresponding pairs of vector elements. The estimated remaining energy of s_j at time t_{k+1} is its accurate remaining energy at time t_k minus the relative energetic distance.

The probability $p(s_j, t_{k+1})$ that sensor node s_j will be selected by s_i as routing next hop toward S is defined as the ratio of the estimated remaining energy of s_j to the summation of the estimated remaining energy of all the nodes in $CNS(s_i)$. Define the *triangular distance* between s_i and s_j with respect to S as $|s_is_j| + |s_js_j^P|$ where s_j^P is the perpendicular projection of s_j on segment s_ic, as shown in Figure 6.11. s_j is assigned a weight $\omega(s_j, t_{k+1})$ which is the ratio of the triangular distance of s_j to the summation of the triangular distance of all nodes in $CNS(s_i)$.

The *weighted entropy* of s_j is defined as $H(s_j, s_i, t_{k+1}) = -\omega(s_j, t_{k+1}) \times p(s_j, t_{k+1}) \times \log p(s_j, t_{k+1})$. It is used to measure the uncertainty of the remaining energy of s_j. The next hop for s_i routing message to S will be an s_j with minimum weighted entropy. In other words, the protocol favors s_j that is closer to s_i and meanwhile closer to the straight line s_ic (thus it is a variant of directional routing). Minimizing $|s_is_j|$ would reduce the transmission energy consumption of s_i; on the other hand, minimizing the distance $|s_js_j^P|$ would minimize the length of the routing path.

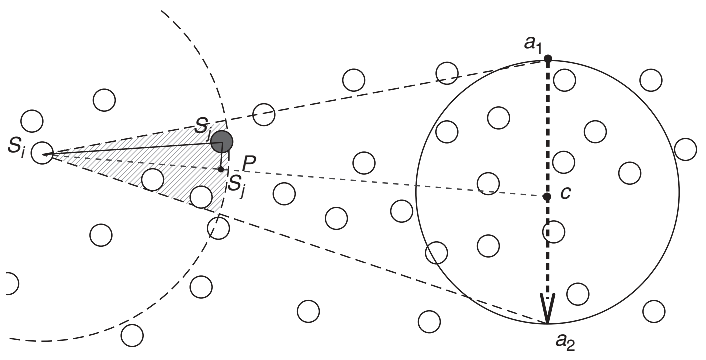


Figure 6.11 WEDAS.

WEDAS has two main drawbacks. While using energetic nodes, it aims at straight line routing with maximized number of hops. This is valid in terms of energy efficiency only under the assumption of $c = 0$ in formula 6.1, which is however not possible in practice. In addition, it has high routing inaccuracy. The reason is twofold. First of all, the routing destination is not the expected sink position, but always the center of the relative mobility zone. Secondly, hop selection is from the CNS restricted by a narrowly defined geographic region. This set may however be empty in the presence of void areas and thus causes routing failures.

Li and Stojmenovic (2008) proposed a localized data dissemination algorithm that improves WEDAS in the above-mentioned two aspects. In this algorithm, the sink distributes its moving speed to all sensors while advertising its relative mobility zone. Using this additional speed information, each sensor is able to estimate the position of the sink at a given time instant, and thus routes messages to the sink. To increase success rate, greedy routing rather than directional routing is used, and the candidate coordinator set is expanded to include those nodes which may be best choices for a possible position of the sink. More specifically, the version of request zone routing presented in Stojmenovic *et al.* (2003) is adopted for source-to-sink data dissemination. This version of request zone routing is discussed in detail in Chapter 8. The concept of weighted entropy is adopted for energy-efficient hop selection. Instead of using energetic distance, entropy is weighted by the inverse of *cost* to *progress* ratio (Stojmenovic, 2006). That is, for node s_j at time t_{k+1}, $\omega(s_j, t_{k+1}) = (|s_i S_{k+1}| - |s_j S_{k+1}|)/|s_i s_j|$, where s_i is current sensor, and S_{k+1} is the estimated sink position at time t_{k+1}. With this new definition of weighted entropy, the algorithm favors an energetic node in hop selection that leads to large progress to the sink and small transmission distance.

REFERENCES

Akkaya K, Younis M, Bangad M. “Sink repositioning for enhanced performance in wireless sensor networks”. Comput Netw 2005;49(4):512–534.

Ammari HM, Das SK. “Data dissemination to mobile sinks in wireless sensor networks: an information theoretic approach”. Proceedings of the 2nd IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS); Washington, DC, USA; 2005. pp. 314–321.

Anastasi G, Conti M, Gregori E, Spagoni C, Valente G. “Motes sensor networks in dynamic scenarios”. Int J Ubiquitous Comput Intell 2007;1(1):9–16.

Anastasi G, Conti M, Monaldi E, Passarella A. “An adaptive data-transfer protocol for sensor networks with data mules”. Proceedings of the 2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM); Helsinki, Finland; 2007. pp. 1–8.

Banerjee T, Xie B, Jun JH, Agrawal DP. “Increasing lifetime of wireless sensor networks using controllable mobile cluster heads”. Wireless Communications and Mobile Computing, 2009. To appear.

Baruah P, Urgaonkar R, Krishnamachari B. “Learning-enforced time domain routing to mobile sinks in wireless sensor fields”. Proceedings of the 29th IEEE International Conference on Local Computer Networks (LCN); Tampa, Florida, USA; 2004. pp. 525–532.

Basagni S, Carosi A, Melachrinoudis E, Petrioli C, Wang ZM. "Controlled sink mobility for prolonging wireless sensor networks lifetime". ACM Wirel Netw 2008;14(6):831–858.

Bentley JL. "Multidimensional binary search trees used for associative searching". Commun ACM 1975;18(9):509–551.

Bi Y, Sun L, Ma J, Li N, Khan IA, Chen C. "HUMS: an autonomous moving strategy for mobile sinks in data-gathering sensor networks". EURASIP J Wirel Commun Netw 2007.

Bogdanov A, Maneva E, Riesenfeld S. "Power-aware base station positioning for sensor networks". Proceedings of the 23th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM); Hong Kong, China, Volume 1; 2004. pp. 575–585.

Bonabeau E, Dorigo M, Theraulaz G. Swarm intelligence: Oxford University Press, New York; 1999.

Buragohain C, Agrawal D, Suri S. "Power aware routing for sensor databases". Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM); Miami, USA, Volume 3; 2005. pp. 1747–1757.

Chakrabarti A, Sabharwal A, Aazhang B. "Using predictable observer mobility for power efficient design of sensor networks". Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN); Palo Alto, CA, USA, Volume 2634 of LNCS; 2003. pp. 129–145.

Friedmann L, Boukhatem L. "Efficient multi-sink relocation in wireless sensor network". Ad Hoc & Sens Wirel Netw 2009. To appear.

Gu Y, Bozdag D, Ekici E. "Mobile element based differentiated message delivery in wireless sensor networks". Proceedings of the 2006 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM); Niagara-Falls, Buffalo-NY, USA; 2006. pp. 83–92.

Gu Y, Bozdag D, Ekici E, Ozguner F, Lee C-G. "Partitioning based mobile element scheduling in wireless sensor networks". Proceedings of the 2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON); Santa Clara, California, USA 2005. pp. 386–395.

Hashish S, Karmouch A. "Deployment-based solution for prolonging lifetime in sensor networks with multiple mobile sinks". Ad Hoc & Sensor Wirel Netw 2009;7(1–2):23–49.

Hwang FK, Richards DS, Winter P. The Steiner tree problem: Elsevier, North-Holland; 1992.

Ingelrest F, Simplot-Ryl D, Stojmenovic I. "Target transmission radius over LMST for energy-efficient broadcast protocol in ad hoc networks". Proceedings of IEEE International Conference on Communications (ICC); Paris, France, Volume 7; 2004. pp. 4044–4049.

Jea D, Somasundara A, Srivastava M. "Multiple controlled mobile elements (data mules) for data collection in sensor networks". Proceedings of the 1st IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS); Marina del Rey, Califonia, USA, Volume 3560 of LNCS; 2005. pp. 244–257.

Kansal A, Somasundara AA, Jea DD, Srivastava MB, Estrin D. "Intelligent fluid infrastructure for embedded networks". Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys); Boston, Massachusetts, USA; 2004. pp. 111–124.

Kim HS, Abdelzaher TF, Kwon WH. "Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks". Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys); Los Angeles, California, USA, 2003. pp. 193–204.

Kruskal JB. "On the shortest spanning subtree of a graph and the traveling salesman problem". Proc Am Math Soc 1956;7:48–50.

Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB. The traveling salesman problem. John Wiley & Sons, Chichester; 1985.

Li N, Hou JC. "Topology control in heterogeneous wireless networks: problems and solutions". Proceedings of the 23th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM); Hong Kong, China, Volume 1; 2004. pp. 232–243.

Li X, Stojmenovic I. "A localized energy-efficient data dissemination scheme for wireless sensor networks with a mobile sink". 2008. In preparation.

Lian J, Naik K, Agnew GB. "Data capacity improvement of wireless sensor networks using non-uniform sensor distribution". Int J Distrib Sens Netw 2006;2(2):121–145.

Luo J, Hubaux J-P. "Joint mobility and routing for lifetime elongation in wireless sensor networks". Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM); Miami, USA, Volume 3; 2005. pp. 1735–1746.

Nesamony S, Vairamuthu MK, Orlowska ME. "On optimal route of a calibrating mobile sink in a wireless sensor network". Proceedings of the 4th International Conference on Networked Sensing Systems (INSS); Braunschweig, Germany; 2007. pp. 61–64.

Nesamony S, Vairamuthu MK, Orlowska ME, Sadiq SW. "On optimal route computation of mobile sink in a wireless sensor network". Technical Report 465. ITEE, University of Queensland; 2006. Available at http://espace.library.uq.edu.au/eserv/UQ:7693/TR-465.pdf.

Olariu S, Stojmenovic I. "Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting". Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM); Barcelona, Catalunya, Spain; 2006.

Prim RC. "Shortest connection networks and some generalisations". Bell Syst Tech J 1957;36: 1389–1401.

Rao J, Biswas S. "Joint routing and navigation protocols for data harvesting in sensor networks". Proceedings of the 5th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS); Atlanta, Georgia, USA; 2008. pp. 143–152.

Rodoplu V, Meng TH. "Minimum energy mobile wireless networks". IEEE J Sel Areas Commun 1999;17(8):1333–1344.

Shah RC, Roy S, Jain S, Brunette W. "Data MULEs: modeling and analysis of a three-tier architecture for sparse sensor networks". Ad Hoc Netw 2003;1(2–3):215–233.

Singh S, Woo M, Raghavendra CS. "Power-aware routing in mobile ad hoc networks". Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom); Dallas, Texas, USA; 1998. pp. 181–190.

Stojmenovic I. "Localized network layer protocols in wireless sensor networks based on optimizing cost over progress ratio". IEEE Netw Mag 2006;2(1):21–27.

Stojmenovic I, Lin X. "Power aware localized routing in wireless networks". IEEE Trans Parallel Distrib Syst 2001;12(11):1122–1133.

Stojmenovic I, Olariu S. "Data-centric protocols for wireless sensor networks". In: Stojmenovic I, editor. Handbook of sensor networks: Wiley, Hoboken, New Jersey, USA; 2005. pp. 417–456. Chapter 13.

Stojmenovic I, Ruhil AP, Lobiyal DK. "Voronoi diagram and convex hull based geocasting and routing in wireless networks". Proceedings of the 8th IEEE Symposium on Computers and Communications (ISCC); Antalya, Turkey; 2003. pp. 51–56.

Sugihara R, Gupta RK. "Scheduling under location and time constraints for data collection in sensor networks". Proceedings of the 28th IEEE Real-Time Systems Symposium (RTSS) Work in Progress Session; Tucson, Arizona, USA; 2007. Available at http://cse.ucsd.edu/rysugihara/papers/rtsswip07.pdf.

Sugihara R, Gupta RK. "Improving the data delivery latency in sensor networks with controlled mobility". Proceedings of the 4th IEEE International Conference on Distributed Computing in

Sensor Systems (DCOSS); Santorini Island, Greece, Volume 5067 of LNCS; Santorini Island, Greece; 2008. pp. 386–399.

Vincze Z, Vass D, Vida R, Vidacs A, Telcs A. “Adaptive sink mobility in event-driven densely deployed wireless sensor networks”. Ad Hoc & Sens Wirel Netw 2007;3(2–3):255–284.

Wadaa A, Olariu S, Wilson L, Jones K, Eltoweissy M. “Training a wireless sensor network”. Mobile Netw Appl 2005;10(1–2):151–168.

Wu X, Chen G. “Dual-sink: using mobile and static sinks for lifetime improvement in wireless sensor networks”. Proceedings of the 1st International Workshop on Distributed Sensor Systems (DSS); Honolulu, Hawaii, USA; 2007. pp. 1297–1302.

Wu X, Chen G, Das SK. “Avoiding energy holes in wireless sensor networks with nonuniform node distribution”. IEEE Trans Parallel Distrib Syst 2008;19(5):710–720.

Xing G, Wang T, Jia W, Li M. “Rendezvous design algorithms for wireless sensor networks with a mobile base station”. Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc); Hong Kong, China; 2008. pp. 231–239.

Xing G, Wang T, Xie Z, Jia W. “Rendezvous planning in mobility-assisted wireless sensor networks”. Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS); Tucson, Arizona, USA; 2007. pp. 311–320.

Chapter 7

Topology Control in Sensor, Actuator, and Mobile Robot Networks

Arnaud Casteigts, Amiya Nayak, and Ivan Stojmenovic

School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, Canada K1N6N5

Abstract

The efficiency of many sensor network algorithms depends on characteristics of the underlying connectivity, such as the length and density of links. It is therefore a common practice to control the number and nature of links that are to be used among all potentially available links. Such *topology control* can be achieved by modifying the transmission radii, selecting a given subset of the links, or moving some nodes (if such functionality is available). This chapter reviews some of these problems, and related solutions applicable to the context of sensor and actuator networks. Spanning structures and minimum weight connectivity are first discussed, and some applications for power-efficient and delay-bounded data aggregation are described. The issue of detecting critical nodes and links to build a biconnected topology is also investigated, with the aim of providing fault tolerance to the applications, and some recent and prospective works considering biconnectivity of mobile sensors/actuators and related deployment of sensors, augmentation, area and point coverage are discussed.

7.1 INTRODUCTION

Unless indicated otherwise, all the solutions presented in this chapter assume that every node is able to know its own position and the positions of its neighbors using exchanges of `hello` messages. These solutions also assume that no obstacles are present and that the network can thus be represented by a *unit disk graph* (UDG), that is, a graph where two vertices are joined by an edge (and called *neighbors*) if and only if the distance between the two corresponding nodes is under a given threshold R. The chosen value for R corresponds to the transmission radius, which is generally assumed to be the same for all nodes. Some variations of this model can however be considered to represent obstacles or different transmission radii among nodes, in which case the edges are considered either as directed (*arcs*), or used only if both communication directions are available. The *min-power* graph is an example of such a nonsymmetrical UDG that represents the cases where each node has the smallest possible transmission radius with respect to the network connectivity. Depending on the scenarios considered, the transmission power can be decided once at the starting time, or adjusted for each message.

The chapter starts with a discussion on the main general approaches used to control the connectivity in static sensor networks. The emphasis is put in particular on the problem of finding minimum transmission radii so that the network is connected (i.e., there exists a path between any two nodes in the network). This problem is actually closely related to the problem of finding a minimum spanning tree (MST), since the longest edge of such structure corresponds to the minimal common radius achieving the connectivity. We thus discuss the problem of approximating this structure in a distributed and localized manner. A few applications of the MST in the context of data aggregation are then presented, with a couple of example protocols that exploit the localized approximation discussed before. Finally, we discuss the problem of maintaining a distributed spanning forest over a uncontrolled sensor and actuator topology so that each sensor belongs to the same tree as at least one actuator.

The second part of the chapter (starting at Section 7.5) is concerned with problems related to biconnectivity. The question of detecting local critical nodes and links in a distributed fashion is first covered. We then discuss several scenarios involving biconnectivity of mobile robots (which may correspond to sensors or actuators, depending on the cases). The first two scenarios address the problem of deploying biconnected sensors around a given point of interest (POI), whereas the two last scenarios address the problem of biconnecting a network that is initially 1-connected.

7.2 GENERAL APPROACHES IN STATIC SENSOR NETWORKS

When the nodes are static, controlling the topology amounts to selecting only a subset of the possible links for effective use according to some desired criteria (distance between nodes, remaining level of energy, avoidance or favoring of cycles, etc.). There are essentially three ways of proceeding:

1. Selecting particular neighbors based on other criteria than the sole distance between them. We refer to Blough *et al.* (2006) as an example of

such selection, where the criterion is to minimize the number of symmetric links so that interferences are bounded. The topology is designed by maintaining the number of neighbors of each node below a value k while guaranteeing the overall connectivity with high probability. The proposed protocol consists of two message exchanges. First, all nodes transmit their ID using the maximal emission power. Upon reception, each node i builds a list L_i containing its k nearest neighbors. Then each node transmits its list using again, the maximum power. A link between two nodes is then kept if and only if it belongs to the k closest neighbors of each other. While the value k needed for connectivity with high probability is logarithmic, their experiments showed that $k = 6$ was the "magic" number above which the network is always connected in practice.

2. Using an adjusted transmission range, possibly being different for each node, and then using all neighbors within that range. While being very energy efficient, the problem of this approach is that *medium access control* (*MAC*) protocols become complicated, and a vast majority of the existing ones, which assume symmetric and same transmission powers, does not work well in this context.
3. Using an adjusted transmission range, but with the same value for every node. This option is often preferred over the previous one, and can also serve as a prior step to further selection. The main question here is to find the minimal value that still guarantees the network connectivity.

We now discuss the construction of a MST, that one can see as being closely related to the first and the third approaches.

7.3 THE MINIMUM SPANNING TREE

MSTs are emblematic example of topological structures commonly used in communication networks. In a graph $G = (V, E)$, an *MST* is a connected subset of the graph that contains all the vertices and minimizes the overall sum of its edge lengths (or more generally their *weight*). A well-known centralized algorithm to build such a structure is the Kruskal's algorithm, which mainly works as follows. All edges of E are sorted in the increasing weight order, and a new empty set E' is created. For each edge e in the ordered set E, e is added to E' if it does not create a loop in $G' = (V, E')$. The resulting graph G' is an MST of G.

The Figure 7.1 gives an example of *MST* that has been built over a random UDG. As usual, in wireless networks, the *length* of the edges was used as their weight. Note that if several edges have the same length, then several equivalent *MSTs* may exist, but this nondeterminism can be easily broken based on additional parameters (such as a comparison between nodes IDs).

An interesting fact about the MST is that its longest edge also corresponds to the optimal common transmission radius (i.e., the minimal radius such that network connectivity is preserved). This follows directly from Kruskal's algorithm, this edge being the last added edge in E'.

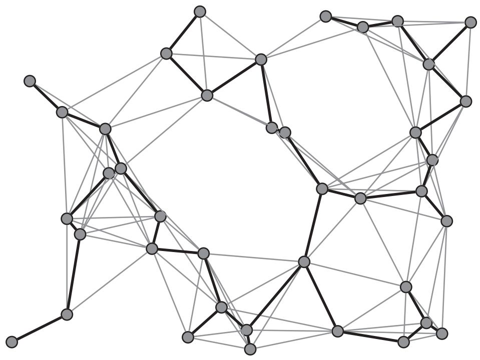


Figure 7.1 A minimum spanning tree on top of a random unit graph.

Considering a few related results, Penrose (1999) proved that the "furthest nearest" neighbor of a node in the network, and the longest MST edge, have asymptotically (when n approaches $+\infty$) the same value. The probability of connectedness exhibits a very sharp transition from 0 to 1 just before the critical value (Bettstetter, 2002). Another interesting fact is that an important amount of energy can be saved if lesser connectivity is required. Santi and Blough (2003) showed for example that, in two and three dimensions, halving the critical transmission range of the nodes still keeps 90% of them connected within a same component.

While the problem of building an MST and finding the optimal transmission range are closely related, both of them require to consider global knowledge such as the size and density, or the spacial distribution of nodes. The problem is then to approximate these optima in a distributed fashion. We present below a few approaches for doing so, based on a localized approximation of the MST.

7.3.1 Localized Approximation of the Minimum Spanning Tree (LMST)

Localized minimal spanning trees (LMSTs), introduced in Li *et al.* (2003) and already discussed in Chapter 2, are distributed structures that approximate the MST using only local information. More precisely, each node u is assumed to collect position information of its 1-hop neighbors, then to compute the local MST that covers itself and its one-hop neighbors (including edges between these neighbors, possibly deduced from their positions). Then, an edge (uv) belongs to the $LMST$ if and only if is belongs to the local MST of both u and v. For illustration purpose, we provide in Figure 7.2 the results obtained using this algorithm, as opposed to the optimal solution given in Figure 7.1. As one can see, just a few cycles (3) were created. Considering that only one-hop information was used, the approximation can be qualified as good.

On the basis of the observation of Penrose that the longest edge of the MST corresponds to the optimal transmission range value, Ovalle-Martinez *et al.*

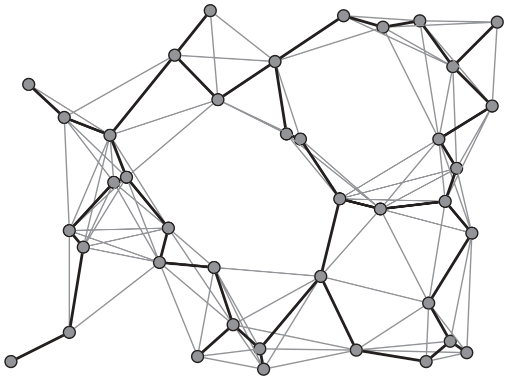


Figure 7.2 Localized minimum spanning tree.

(2005) proposed to use the LMST algorithm to find this value. More precisely, the basic idea was to find the longest LMST edge (that approximate this value) using a wave propagation within. However, it was observed in this paper that even a very small number of additional edges (such as those creating a cycle in Figure 7.2,) representing less than 3% in their experiments (on networks with up to 500 nodes) may extend the range value by about 33%. In turn, this 33% of range value may represent an increase of 50% or more in energy consumption, depending on the power attenuation factor that is considered. A quasi-localized scheme is thus proposed in the paper to remove additional edges of the *LMST*, using, in average, less than seven messages per node. This procedure is a *loop breakage procedure*, which iteratively follows dangling edges from leaves to loops. The loops are then broken by eliminating their longest edge. These procedures continue until they all end up at a same node, considered as a *de facto* leader, which can learn the value of the longest edge in the process, and broadcast it back to the other nodes. This procedure operates only in two dimensions, since it is based on a face routing scheme (see Chapter 4 for details on face routing).

7.4 DATA AGGREGATION

A common scenario of sensor networks involves the deployment of hundreds or thousands of low-cost, low-power sensor nodes in a region from where information will be collected periodically. Sensor nodes must sense their nearby environment and send the information to a sink or actuator, from which the collected information can be further processed or made available to the user. In the most basic reporting schemes, each sensor independently sends its data to the associated actuator using routing operations that generates a lot of redundant traffic if the data is geographically or temporally correlated for example, if two neighboring, or successive, measure values are expected to be close to one another). *Data aggregation* arises from the observation that most of this redundancy could be avoided if the data were partially processed locally to the sensors, for example, by averaging it over time or space before forwarding it.

This is what *data aggregation* does, by applying fusion/consolidation functions to the data along its way to the sink. Example of such functions include *average, maximum, minimum, sum, count*, or *deviation*, that can be applied either periodically or on-demand.

Once these functions are chosen and combined, the main problem of data aggregation is to build the overlay structure along which data will be effectively aggregated. This structure must be as efficient as possible to allow a fast aggregation while maximizing the lifetime of the network (i.e., the number of aggregation cycles or *rounds*, before energy depletion).

A very common type of structure in this context is the tree, which represents a natural hierarchical organization where parent nodes collect and aggregate the data coming from their children, before forwarding their own data in turn. At the top of the tree is the sink (or actuator). Figure 7.3 gives an example scenario where sensors are reporting toward four actuators. Here the trees are set up in such a way that the sum of edge lengths between sensors and actuators are minimized. Other criteria such as minimizing the number of *hops* or favoring the nodes with more remaining energy could be used instead.

In order to be realistic and efficient in the context of sensor and actuator networks, an algorithm to build such a structure should have some important properties. First, the algorithm should be distributed since it is extremely energy consuming to calculate the optimum paths in a dynamic network and inform others about the computed paths in a centralized manner. The algorithm must scale well with increasing number of nodes. Another desirable property is robustness, which means that the routing scheme should be resilient to node and link failures. The scheme should also support new node additions to the network, since not all nodes fail at the same time, and some nodes may need to be replaced. In other words, the routing scheme should be self-healing. The final and possibly

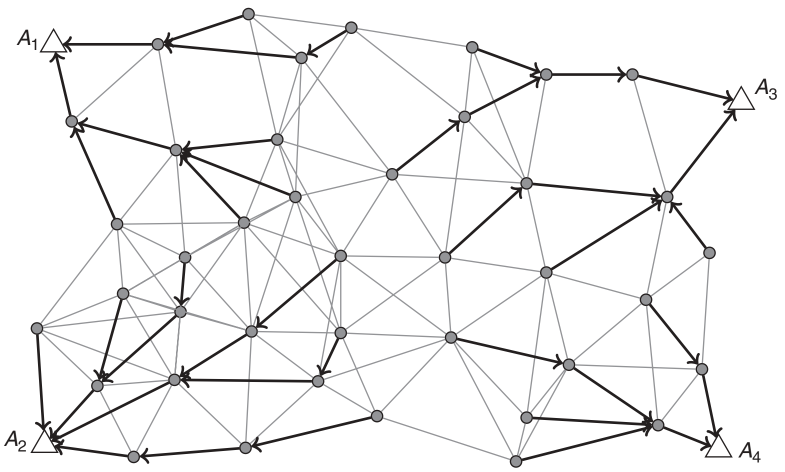


Figure 7.3 Data aggregation trees based on the cumulative distance to the sink.

the most important requirement for a data aggregation scheme for wireless sensor networks is being energy efficient.

One common approach to build an aggregation tree is to flood a packet from the sink (or actuator) so that every node can select a parent among the nodes from which it received the packet. The data is then aggregated along this so-constructed hierarchy. The problem with flooding techniques (some of which are detailed in Chapter 2) is that this may generate a lot of redundant messages, be problematic with respect to possible interferences, and more generally, cost a significant amount of energy, since several retransmissions happen uselessly. In Tan *et al.* (2007), the idea to build and maintain an underlying *LMST* was proposed, which is considered as the topology during the flooding operation, so that the overhead of messages is much reduced (the same principle may also work with other structures such as the *relative neighborhood graph*, as pointed out by the authors).

A set of three protocols, called *localized power-efficient data aggregation protocols* (L-PEDAPs), are proposed. Their variations correspond to different strategies of parent selection. According to the decisions made during this flooding process, the tree is yielded. The three methods are: (i) choosing the first node from which the flooded packet is received, (ii) choosing the node that minimizes the number of hops to the sink, and (iii) choose the node that minimizes the total energy consumed over the path to the sink. Note that the first and second methods are nearly equivalent in sensor networks, since the processing time of messages overcomes their physical transmission time over the air, making the transmission time quasi-proportional to the number of hops.

Since the underlying structure (here, the *LMST*) can be locally maintained, this solution accommodates new node arrivals and departures of existing nodes. The authors also propose power-aware versions of these protocols that consider the energy level of nodes while constructing the underlying structure. Finally, the paper derives a theoretical upper bound for the lifetime in terms of the first node failure, and simulation results show that the protocols achieve up to 90% of this upper bound.

7.4.1 Delay-Bounded and Power-Efficient Data Aggregation

While the purpose of the LMST is to build an energy-efficient structure by selecting the shortest edges, it is not optimal with respect to delay considerations. In Xu *et al.* (2009), the authors propose a data aggregation protocol where a LMST is first considered, then modified at reporting time in order to match additional delay constraints. As with most existing solutions for delay-bounded data aggregation, the metric used here to approximate the delay of a communication is the number of hops. Indeed, among the delays experienced by a packet at each hop, the distance traveled over the air is negligible compared to the time required to process it at the nodes, which can be considered equal for each node when all packets are of the same size. The delay experienced by a packet is thus directly proportional to the number of hops it travels. The proposed algorithm, called *desired hop progress* (DHP), allows to respect given delay bounds while using significantly less energy than the known competitor in Melodia *et al.* (2005) (from 25 to 75% less energy consumption and up to 123% network lifetime depending on the configurations). We detail it now.

As with the previous algorithm (L-PEDAP), a tree is built on LMST edges when an actuator (they can be several) floods the network with a request. Thanks to a specific retransmission mechanism, every node computes and memorizes several parameters during this process, the main of which is the hop distance to the sink. Two kinds of distances are actually memorized: the *localized minimal spanning tree-based* distance (LD), which represents the number of hops to the actuator using only the LMST edges, and the *unit disk graph-based* distance (UD), which is the number of hops to the actuator if any edge could be used. These different distances are illustrated in Figure 7.4, where thick edges represent the LMST-based tree, and all edges together, the UDG. The aggregation process then starts from the leaves to the actuator as explained below.

Every request (packet flooded) contains information about the delay that must be respected during this aggregation. If this delay is achievable using only LMST edges (that is, if for every node, LD is equal to, or lower than the desired delay), then the aggregation occurs as with L-PEDAP using only these edges. Otherwise, the reporting nodes try to find *shortcuts* outside of the LMST, using the UDG distance. More precisely, before sending the aggregated packet to its parent, every node calculates a particular ratio to decide whether using LMST edges is sufficient or a shortcut is necessary. This ratio, called dhp, is defined as:

$$dhp = \left\lceil \frac{LD}{Delay - MED} \right\rceil, \tag{7.1}$$

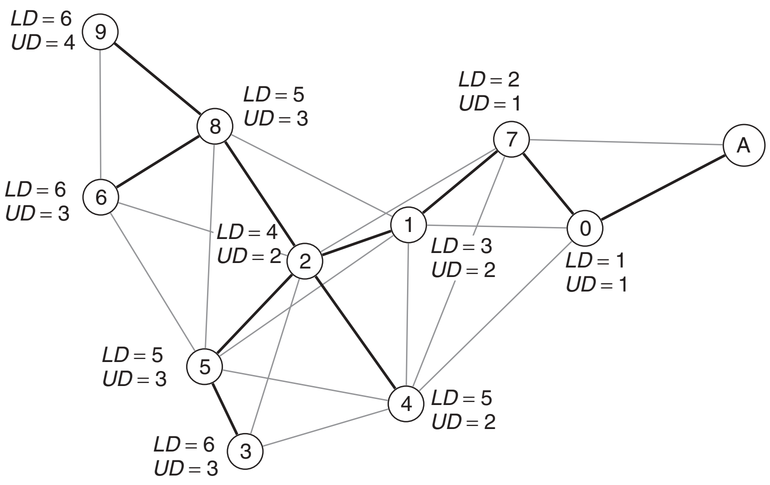


Figure 7.4 Example topology for the DHP protocol.

where MED (*most experienced delay*) is the number of hops already done by the children. The meaning of this ratio is actually to represent the LD distance that should be gained at every next hop so that the delay is finally matched. If this value is higher than 1, then it determines which level of shortcut should be ideally used. Considering the scenario given in Figure 7.4 with a delay limit of 3, this leads to the following execution.

Since node 8 has a UD equal to the required delay, it cannot accept any child (because the delay could not be respected whatever the shortcuts). Node 9 will therefore not be considered for the aggregation. Since nodes 3 and 6 have both a *dhp* value of 2 ($\lceil \frac{6}{3-0} \rceil$), they try to apply a shortcut to gain two LMST hops, and thus select node 2 as parent. Nodes 4, 5, and 8 compute their *dhp* (also equal to 2), and thus select node 1 (shortcutting node 2). Node 2 then computes *dhp* (also equal to 2), and takes node 7 as parent. For the same reasons again, node 1 decides that node 0 is its parent. In turn, node 7 finds that its report has to be sent directly to the actuator, as a single last hop is allowed ($Delay - MED = 1$). Finally, node 0 naturally determines the actuator to be its parent. While almost no LMST edges were used in this example, the point of this protocol is that such edges are always used when the delay constraint allows it, which tends to offer both delay-bounded and power-efficient aggregation at the same time.

Two variations of the DHP protocol were also proposed in the same paper (DHPA and DHPAC), to integrate it with *sensor activity scheduling* and *connected dominating set* (CDS), respectively. Detailed discussion on CDS can be found in Chapter 1. The two variants require fewer sensors to report and thus have reduced bandwidth usage and improved energy efficiency. The first variant, DHPA, adopts a localized area coverage algorithm (Gallais *et al.*, 2006) for selecting an active node set. Active nodes monitor the environment and generate reports, whereas the others switch to sleep mode for energy saving. The DHP protocol is therefore run only on active nodes. In this area coverage algorithm, each node sets a time-out t to start coverage evaluation and schedule its activity. Considering that nodes with shorter t will have a higher chance to stay active, they define $t = c/E_{\text{rest}}$ (c is a constant, and E_{rest} is remaining nodal energy). This definition favors energetic nodes. That is, the more residual energy a node has, the more chance the node gets to work. The second variant, DHPAC, is a combination of DHPA and the localized CDS algorithm from Carle and Simplot-Ryl (2004). In this combination, the CDS algorithm is run on active nodes determined by the area coverage algorithm. Since each active node either belongs to the CDS or has a direct neighbor in it; non-CDS nodes will report to their closest CDS neighbor, while CDS nodes will run the DHP protocol to organize data aggregation within the CDS.

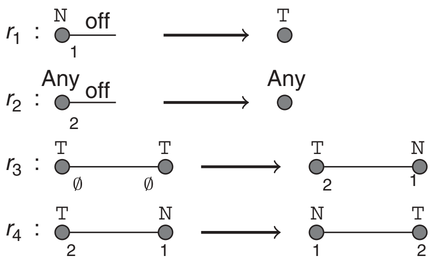
7.5 SPANNING TREES IN UNCONTROLLED DYNAMIC TOPOLOGIES

The discussion below addresses the scenarios where sensors are to move in an uncontrolled fashion, which is for example the case when they are carried by some physical actors of the considered scene (e.g., animals, vehicles, virtual insects, or robots whose movements are to be determined by external parameters). The problem of maintaining distributed spanning trees over dynamic topologies has been extensively studied these past few decades, especially in the two research areas of dynamic graphs and mobile *ad hoc* networks. To the best of our knowledge, a vast majority of approaches, if not all of them, considered the problem of building a *single* spanning tree to cover the whole network. Generally, tackled from the angle of *self-stabilization*, these approaches consider topological events as *faults* that induce a *nonlegal state*, which the algorithm must correct. The correction is then achieved when the whole network is covered by a single tree, which is the *legal state*. To give a few references on this family of approaches, one can cite the distributed graph algorithm in Awerbuch *et al.* (1993), which exhibits the shortest construction time "from scratch" [recently adapted to the message passing model in Burman and Kutten (2007)], and Gaertner (2003) that describes a number of comparable approaches.

While most proposed algorithms engage a complete reconstruction of the tree after each topological failure, some other more realistic approaches [e.g., (Baala *et al.*, 2003), (Abbas *et al.*, 2006)] attempt to correct only the local discrepancies resulting from the link failures. However, these algorithms still require some stabilization time during which the separate subtrees are unavailable and inconsistent. An important consequence is that they simply cannot deal with topologies that change quicker than the stabilization time.

Up to now we discussed scenarios where the connectivity of the whole network was required. This might not always be the case, however. Most applications for sensor and actuator networks do not actually require that a path exists between a sensor and all of the actuators, the point being that sensor must only be able to report information to, or communicate with, at least *one* actuator (or perhaps a few, for fault tolerance). On the other hand, is it of utmost importance that such communication is *always* achievable, that is, the underlying supporting structure is always available.

A novel approach addressing highly dynamical topologies was proposed in a recent paper (Casteigts *et al.*, 2009). The basic idea behind this approach is to renounce building a single tree covering the whole network, and instead consider maintaining a forest of several trees that grow opportunistically and recover a consistent state in one single operation after any topological failure in such a way that the two parts of a given broken tree remain transparently usable. The key point to achieve such property is that both *mergings* and *splittings* of the trees are *purely* localized events that do not generate any wave propagation. The algorithm relies on the circulation of tokens whose number is strictly maintained at one per tree. The difference with other token-based approaches is that the walk of each token is limited to the edges of its tree, which offers some very specific properties. The primary property is that every node knows at anytime, which one of its local edges leads to the token, this edge being simply the one through which the token went out after its previous visit. The consequence is that when an edge of a tree is broken, one of the two end point nodes knows



Initial states:

- `T` for every vertex
- ∅ for every edge extremity

Meaning of the states:

- `T`: a token is on this node
- `N`: no token is on this node
- `1`: this tree edge leads to the token
- `2`: this tree edge does not

Figure 7.5 The spanning forest algorithm.

that its remaining part of the tree is token free, and that it is now the "highest" node on the route that led to the lost token. As a consequence, it can locally regenerate a new token and resume the circulation transparently for the other nodes.

The principle of this algorithm can be detailed by a small set of localized modification patterns, depicted in Figure 7.5. Initially, every vertex is a one-vertex tree that has its own token (label `T`). When two tokens are located on neighboring vertices, they are merged and the corresponding edge is marked as a *tree edge* (rule r_3). This marking use a different value on each side (`1` and `2`) to reflect the orientation induced by the remaining token. If no merging is locally achievable, the token is transmitted to any neighbor *in* the tree (rule r_4), and the orientation mark is updated consequently. For any given node, if the local edge leading to the token is broken, then a new token is regenerated locally (rule r_1). The other side of the broken edge does not perform any particular operation (rule r_2).

An interesting question in the context of sensor and actuator networks is whether the expected size of the trees is large enough to guarantee that each sensor has at least one actuator in its tree at anytime. In other words, one might want to answer the following question: "*given a number of sensors, their density and the expected rate of topological changes, how many actuators are needed so that the probability, to have at anytime at least one actuator in each tree, is above a given threshold?*". In Casteigts *et al.* (2009), the authors provided a first element of answer by characterizing the expected merging time of two given trees, as a function of their size and the number of links available between them. The road between these first results and the complete answer may still be important, though.

7.6 DETECTION OF CRITICAL NODES AND LINKS

In sensor and actuator networks, the failure of some nodes or links, if generating several partitions of the network, may be fatal for collecting data from the field or performing certain actions on sensors. It is expected, however, that the network exhibits some *critical connectivity* before partitioning. Recognizing such properties in a timely manner could allow to perform some data or service replication, so that the network can continue to function after the partition occurred. This kind of detection may also be used at deployment time (e.g., while deciding a common communication radius) to ensure that no such critical node or link exists, that is, the network is *biconnected*. Both approaches may be considered to add fault tolerance to the network. We discuss below some ideas for detecting critical links and nodes.

Algorithms for detecting critical nodes and links based on global knowledge are well known. However, their use in sensor networks is limited since this requires the entire topology to be known by a single entity, which is not scalable and implies a delay between topological changes and the system reactions. It appears therefore preferable to try to detect critical links and/or nodes in a local and distributed manner, even if making possibly a few appreciation mistakes that imply a more "pessimistic view" of the connectivity (an element seen as critical while not being so).

In a global context, a node or link is said to be critical, if its removal disconnects the network, that is, if this partitions it into several connected components. The definitions of what critical nodes and links are must be slightly modified in a localized context. As introduced in Jorgic *et al.* (2004), a node can be said to be *locally critical* if its removal disconnects the subgraph of its p-hop neighbors. In the case of links, several definitions can be considered, and three were proposed in the same paper, based on the method used to look at the local connectivity. These methods are illustrated in Figure 7.6. The first method consists of looking at the p-hop neighbors of both end points and see if some are in common (Fig. 7.6a). The second one is to initiate a face traversal on both sides of the tested link in order to see if the other end point is reached before two hops (Fig. 7.6b). Finally, the link can also be decided critical if its end points are critical themselves, and this information is already available (Fig. 7.6c).

Depending on whether position information is available to the nodes, a variation of each definition can be considered, as discussed in Jorgic *et al.* (2004). Let us consider the simple example given in Figure 7.7, where nodes A and B must determine if their common link is critical, based on a one-hop neighboring

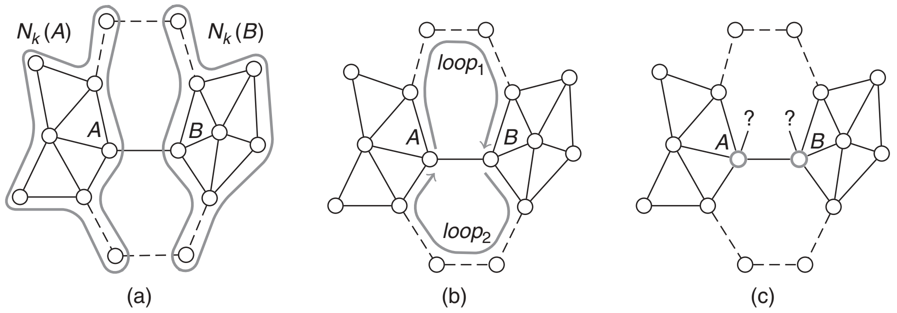


Figure 7.6 Localized detection of a critical link (here (AB)). For each method, the link is decided critical if the caption formula is true.

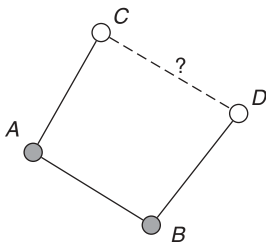


Figure 7.7 Impact of the availability of position information on the detection of criticality.

information. If the respective positions of C and D are known, then the fact whether a link exists between them or not can be established without additional information, while this is not possible using only topological information. Obviously, such position-based deduction assumes the UDG model without obstacle. These two variations are denoted as k-*top*- and k-*pos*-criticality by the authors.

Experiments using random UDGs showed a high correlation between global and local decisions. The only difference is when alternative routes exist, but are relatively long. The point is that really critical elements will be detected as such, and any wrong appreciation is only generated by excess of caution (in case of reactive replications) or a slight excess of connectivity (in case of link selection).

These notions can be generalized to the case of critical k-connectivity of the network, as proposed in Jorgic *et al.* (2007). In the first protocol of this paper, each node makes a criticality decision by verifying whether or not each of its p-hop neighbors has a degree (number of neighbors) of at least k. The second protocol tests also whether the subgraph of the p-hop neighbors of a given nodes is k-connected. The third protocol also verifies whether this subgraph contains any critical nodes.

7.7 BICONNECTED ROBOT TEAM MOVEMENT FOR SENSOR DEPLOYMENT

We consider here, the problem of deploying static sensors around a POI using a fleet of mobile robots able to carry them. We describe the solution proposed in Li (2009). Here, the number of mobile robots is considered arbitrary (and limited), and each one is initially supplied with an arbitrary number of sensors. The problem is then to deploy collaboratively the sensors so that their topology forms a *triangle tessellation* around the POI. The choice of a triangle tessellation is motivated by its interesting geometrical properties that create a near-optimal coverage by the sensors while making them biconnected as a by-product. The main concern is to ensure that the robot network also remains biconnected during the deployment, while minimizing the sum of their moves.

The principal steps of the proposed protocol are as follows. At the beginning, the robots are randomly scattered in the region. They first run an auxiliary protocol such as *greedy-rotation-greedy* (see Li *et al.*, 2007 or Chapter 10) to gather around the POI in several concentric hexagonal layers that form a triangle tessellation at the local scale. Let us first assume that the number of robots is such that the outermost layer is complete (as with the seven-robots hexagon depicted in the middle of Fig. 7.8). Each robot starts by dropping one sensor at its position; then the whole group of robots shifts in one direction, and starts a circular (or more precisely, a hexagonal) course around the already deployed sensors, dropping new sensors along their way. This process repeats until all sensors have been deployed. Note that each circumvolution can deploy several layers, depending on the group diameter.

Let us call *frontier robots* the robots that are located in the front/head of the group with respect to the current direction (i.e., robots 2, 1, and 6 on the hexagon representing the group after the initial shift). Frontier robots are those in charge of dropping their sensors when they encounter an empty virtual vertex of the tessellation. Whenever the group arrives at a corner of the hexagon, the frontier nodes change according to the new direction (in the same example, the new frontier robots after the first corner has been reached will be robots 1, 6, and 5). If the frontier robots run out of sensors at some point, the second layer of robots (e.g., 5, 0, and 3 between Corner 0 and Corner 1) will take over the task. It is however expected that a reorganization of the robots within the group may be required in some situations, especially if the initial supply in sensor is not uniform among robots.

The problem becomes more complex when the number of robots does not correspond to a perfect hexagon. Some sketches of solutions given in Li (2009) are depicted on Figures 7.9a and 7.9b. The first picture corresponds to the case where the outermost layer contains only one robot. It is suggested that robots in the inner hexagons behave as previously, while the robot in the outermost layer rotates around the group when a corner is encountered (this robot is expected to be relatively quickly depleted in sensors, thereby forcing more frequent reorganizations of the group). The second picture corresponds to cases where the outermost hexagon contains more than one robot. In this case, the robot team is to move using a different pattern. Here the team is a hexagon with two robots on the sides of the frontier. The other extra robots can be placed around the core, for example, at the positions marked by empty circles. As for the previous case, the outer robots will have to move within the group when a corner is encountered.

7.8 AUGMENTATION ALGORITHM FOR ROBOT SELF DEPLOYMENT

In the same vein as the previously discussed scenario, the deployment of a biconnected network of sensors around a POI is addressed in Falcon *et al.* (2009). The major difference is that sensors are themselves endowed with movement capabilities, and are therefore capable of self-deploying around the POI. Here, the sensors are released one at a time from potentially different remote places.

The main idea of the proposed protocol is to incrementally build a perfect triangle tessellation around the POI (this tessellation structure is chosen here for the same geometrical reasons as previously discussed), while minimizing the sum of sensors moves. This protocol roughly works as follows: the first sensor moves

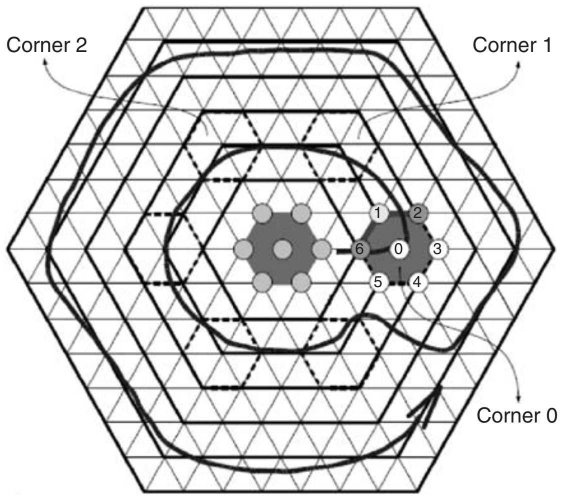


Figure 7.8 Simple case of deployment by mobile robots.

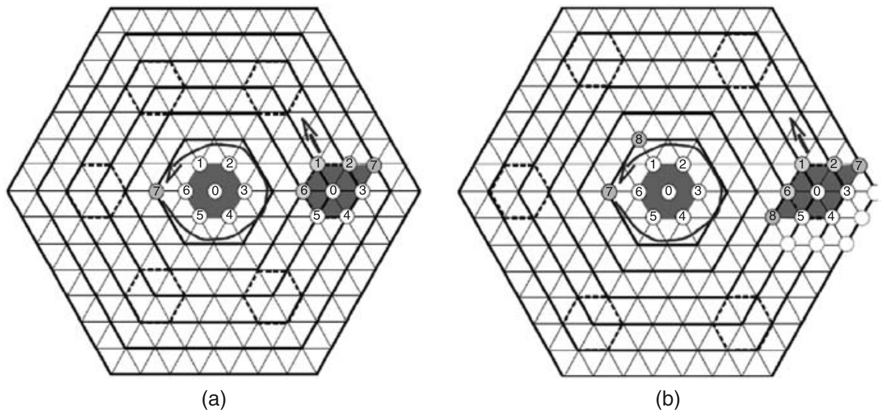


Figure 7.9 Robot team behavior. (a) One robot in the outmost layer. (b) Several robots in the outmost layer.

directly to the exact position of the POI. Then, when a new sensor is released, it moves toward the POI until entering the range of a sensor that already belongs to the tessellation. At this point, the already deployed sensor is in charge of finding an appropriate tessellation position to ask the new node to move at.

The main objective here is to minimize the sum of robots' movements while keeping the tessellation centered around the POI. The kind of choice resulting from these constraints is illustrated in Figure 7.10, where an empty tessellation vertex is available at the opposite of a newly arriving node (node N). Here, we should prefer to shift every node in the diagonal instead of asking the new node to turn around the tessellation (because this latter movement represents a much larger overall distance). However, such decision implies that a tessellation node

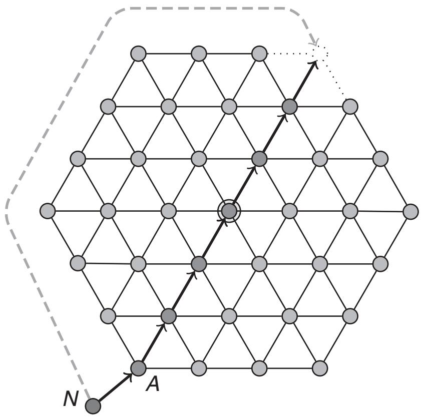


Figure 7.10 Minimizing the sum of movement to place a new node.

can be aware of, or inquire for, an empty remote position in the tessellation. As forcing all nodes to memorize (and synchronize) a global view of the tessellation is not reasonable, the challenge will be to design a distributed protocol where nodes can collaboratively decide what destination can be assigned to a new node. Note that the question of how uniformly the movements are distributed among robots may arise as a second step, since here the already deployed nodes can be asked to move again afterward. Another interesting result could be to characterize an upper bound on the deploying time of one sensor, in order to determine how frequently they can be initially released (this time might increase with the number of sensors previously deployed).

7.9 BICONNECTIVITY FROM CONNECTIVITY WITHOUT ADDITIONAL CONSTRAINTS

We now discuss some scenarios where mobile robots were already randomly deployed but still assumed one-connected. From this initial connected network, the objective is to turn the network biconnected using only localized movement decisions and minimizing the total movements of robots. The solution presented in this section comes from Das *et al.* (2009).

From a *global* point of view, a biconnected network is a network that does not contain any *critical* nodes nor *critical* links, that is, that remains connected if any node or link is individually removed. Since a link is critical only if at least one of its end point nodes is critical, making the network biconnected comes to turn every critical node into noncritical. By looking at the Figure 7.11, it appears intuitively clear that the *global* criticality of a node (e.g., N) cannot be locally decided, since it depends on some remote edges (e.g., AB), whose existence is locally unknown.

The algorithm is based on the concept of *p-hop criticality*, already discussed in 7.6. More practically, each node is assumed to collect information about its p-hop neighbors through exchanging and relaying *hello* messages over multiple

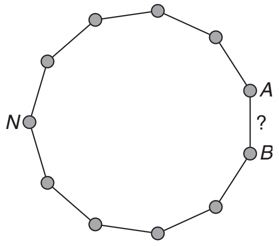


Figure 7.11 Locally critical node versus globally critical node.

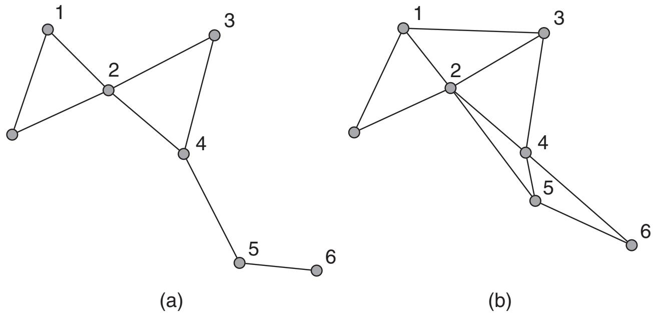


Figure 7.12 Simple scenario for the algorithm from citation for Das *et al*. (2009). (a) Initial network. (b) Resulting network.

hops. If the p-hop neighborhood of a node n appears to n as disconnected without itself, then n locally decides that it is *p-hop critical* (we will simply say *critical* in the following text). In the example depicted in Figure 7.11, N will decide that it is critical (unless $p \geq 5$).

By definition, the movement of any critical node is susceptible of disconnecting the network. The basic idea of this protocol is thus to use only noncritical nodes to create biconnectivity, while keeping the critical nodes *static* (until they become in turn noncritical and able to move). The algorithm is thus based on a small set of predetermined actions that critical nodes trigger on their noncritical neighbors according to the situation. In particular, the fact that a critical node has, or has not, other critical neighbors will generate different actions. Let us define a few more concepts before detailing these actions.

A critical node is called *available* if it has *at least one* noncritical neighbor, in other words if it has a neighbor that can move. This notion of *availability* can be used to decide which one among some critical neighbors is to pilot the local actions. More precisely, the pilot node in this case (also called *critical head*) must be *available* and have a *larger* ID than any of its *available* critical neighbors (in case of tie). Considering the example in Figure 7.12a, nodes 2, 4 and 5 are all critical. Since 4 is larger than 2 and node 4 is available, node 2 is not a critical head. Since 5 is larger than 4 and 5 is available, node 4 is not a critical head neither. Node 5, here, is the only critical head. Note that if node 2 had a larger ID than node 4, there would have been two critical heads here instead of one (nodes 2 and 5).

The algorithm works as follows. At the initialization stage, each node checks whether it is a p-hop critical node, and will continue to check it after every *hello* message exchanged. Whenever a node detects that it is p-hop critical, it broadcasts a *critical announcement* packet to all its direct neighbors, including the information about its *availability*. Two cases are then possible:

1. If the node has no critical neighbor, then it selects two of its neighbors n_1 and n_2 that belong to separate "biconnected components" and asks them to move toward each other. If the distance between n_1 and n_2 is d, then each node should move a distance of $(d - r)/2$, where r is the communication range. In case of several possible choices, the pair minimizing d is chosen.
2. If the node has one or several other critical neighbors, then it figures out whether it is a critical head or not, and if so, it asks a noncritical neighbor to move toward one of the other critical neighbors. Here, the selected neighbor should move a distance of $d - r$. As in the previous case, the pair is selected so that both nodes belong to separate "biconnected components" with the distance d being as small as possible.

If a node receives a request of movement while being already in the process of moving, it simply ignores it; if it receives several requests at the same time from different neighbors, it considers the request coming from the one with the largest ID. Note that the resulting movements may thus not create a new link at every expected place, which is the case for example, when a node moves toward another, which finally moves elsewhere (in case 1). However, since the algorithm is incremental, this situation is likely to be solved at a later iteration.

Considering again the network given on Figure 7.12, we will describe the execution sequence transforming the initial topology (Fig. 7.12a) into a biconnected topology (Fig. 7.12b). As explained above, nodes 2, 4, and 5 are initially critical, but only node 5 is a critical head. It is thus the first node to act by asking node 6 to move toward node 4. Then node 5 becomes noncritical and node 4, which has become a critical head, asks node 5 to move toward node 2. Finally, node 2 remains the only critical node, and applies the first case by asking nodes 1 and 3 to move toward each other, after the network is biconnected.

This algorithm was experimentally compared to the centralized algorithm (Basu and Redi, 2004). Simulation results showed that the total distance of movement of robots is significantly lower with the localized algorithm (about 2.5 times for networks with density 10). However, this algorithm does not totally guarantee biconnectivity, and may even disconnect the network in some particular cases (e.g., when two noncritical nodes happen to be in-between some separate components, and they are both asked to move simultaneously away from each other).

7.10 BICONNECTIVITY FROM CONNECTIVITY WITH ADDITIONAL CONSTRAINTS

We are interested here, in the same problem as in the previous section, that is, to achieve biconnectivity of mobile robots starting from a random (but connected) topology. Here however, the objective is also to maximize the overall coverage and minimize the network diameter at the same time. Every robot n is assumed to have a *communication* range, and a *coverage* range. The first, denoted $c(n)$, indicates up to which distance other nodes can receive messages from $c(n)$. The second, denoted $s(n)$, is the radius defining the area where the robot is to serve. For example, if n is a sensor, then $s(n)$ corresponds to its *sensing* range. If n is an actuator, then $s(n)$ may correspond to the area in which sensors are monitored by n. The ratio between these ranges is usually considered to satisfy $cr(n) > 2 \times sr(n)$. The *overall coverage* of the network is defined as the *union* of the coverage areas of all the robots. One can intuitively see that the closer the robots, the smaller the overall coverage due to potential overlappings.

The *diameter* of a network is defined as the “largest” shortest path between any two nodes. More formally, if $d(u, v)$ is the length of the shortest path between two vertices u and v, then the diameter of a graph $G = (V, E)$ is defined as $max(d(u, v) : u, v \in V)$. Because the diameter bounds the number of hops of transmissions, making it small is crucial for most real-time applications. It can be intuitively seen that the closer the robots, the smaller the diameter of the network. Hence, the concepts of *diameter* and *overall coverage* appear somehow antinomic and may present contradictory objectives for an algorithm.

The Figure 7.13 illustrates this point with a topology of four biconnected robots. In Figure 7.13, the diameter has been reduced in the extreme (1), which generates a substantial overlap of the coverage areas. If we take the same network and move the nodes away from each other until their coverage areas do not overlap (Fig. 7.13b), then the diameter of the network increases (here to 2). However, as depicted in Figure 7.13c, some geometrical organizations such as the *triangle tessellation* seems to offer an interesting trade-off between diameter and coverage.

Hence, it appears a good option to consider this pattern of organization. However, since we consider here, a scenario where the nodes are initially already

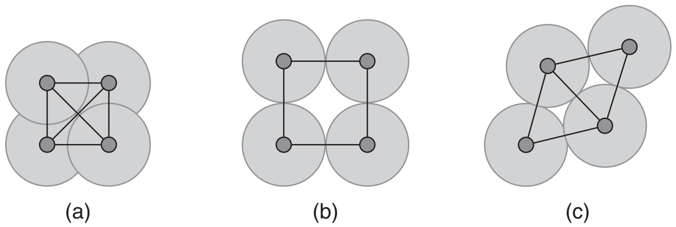


Figure 7.13 Three configuration examples. (a) Maximum diameter. (b) Maximum coverage. (c) Triangle tessellation.

deployed, we do not want to build such a perfect topology over the whole network (but rather use it to determine local organization of neighboring nodes). Let us nonetheless continue with some ideas illustrated on a whole tessellation. Using such structure, regulating the trade-off between *coverage* and *diameter* can be done my merely tuning the distance between neighboring nodes without changing the organization geometry (the smaller distance, the higher priority for a small diameter). This is illustrated in Figure 7.14, where the Figure 7.14a illustrates the choice to favor coverage (with a diameter of 3), and Figure 7.14b one the choice of reducing the diameter to 2 (but with overlapping coverages). Because the only difference between favoring coverage and diameter lies in the distance between nodes without modification of the topology geometry, this suggests that part of the trade-off can be achieved using only localized operations (e.g., determining the local distance to direct neighbors).

The algorithm presented in the previous section (Das *et al.*, 2009) can be adopted so that the resulting network tends to have a smaller diameter and/or a higher coverage depending on a given trade-off parameter. This adaptation could also apply localized triangle tessellation pattern that helps improving both criteria at the same time.

As a by-product, the original algorithm already reduces the diameter of the network by moving the robot closer to one another. This aspect could however be further improved by changing the way the nodes move. For example, instead of simply asking two nodes to move half the distance toward one another to get connected, one could first check if one of them has no additional neighbor, and then ask only this one to move. More generally, making the nodes with lower degrees move more than the others could help decrease the network diameter (although it could also raise some new problems at the same time).

Once the network becomes biconnected, a kind of localized triangle tessellation could be achieved by using repulsive forces, pushing the nodes away from one another equally, and therefore increasing the coverage. Here, the range of the force would serve as the trade-off parameter between coverage and diameter. The Figure 7.15 shows a simple scenario where such forces were applied after the original algorithm. These forces may be applied once the first algorithm terminates, or they could be merged with it.

The process of applying repulsive forces should however carefully avoid to *bi*-disconnect the network. Note that the use of virtual forces, introduced in Zou

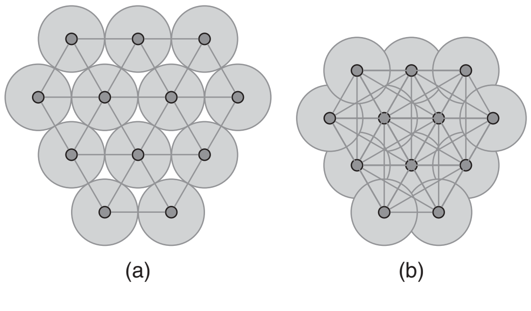


Figure 7.14 Different choices of trade-off. (a) Favoring the coverage. (b) Favoring the diameter.

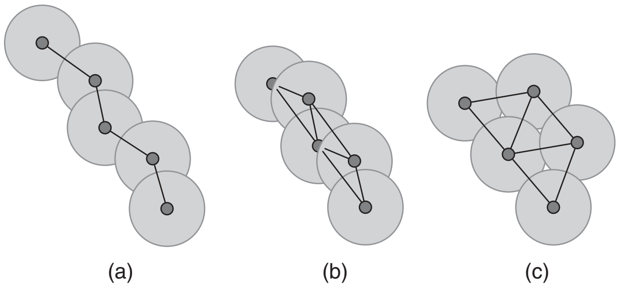


Figure 7.15 Mixing biconnectivity and repulsion forces. (a) Initial topology. (b) After DLNS. (c) After repulsion.

and Chakrabarty (2003) in the context of mobile sensor networks, was recently used to maximize the overall coverage of a robot network.

In Guang *et al.* (2008), the authors consider an initial network that is not necessarily connected and propose an algorithm to make it connected. It is assumed that all the nodes know a common location (e.g., the base station) toward which they can move. Each robot that arrives at this location (e.g., in range with the base station) floods a packet so that every node that receives the packet is thus aware of being connected with the others. Once all nodes are connected, virtual forces are applied to maximize the coverage of the network.

In Liu *et al.* (2009), a localized protocol is proposed to biconnect a network of robots from an initially nonconnected topology. The objective is to minimize total moving distance of robots while maximizing sensing coverage of the network. It assumes that robots have a common communication range and a sensing range. Each robot is aware of locations of one-hop neighbors and the boundary of the sensing region. As with the previous solution, all robots are supposed to move toward a common position and maximize their sensing coverage. Here, however, nodes move by following only two kinds of virtual forces. The first is an attraction force that always draws every robot toward the common POI, and the second is a repulsive force that is applied between every pair of neighboring nodes. The simultaneous application of both forces (illustrated on one node in Figure 7.16 ends up in a biconnected triangle tessellation centered on the POI (such as the one on Fig. 7.17, which results from effective simulations). This is due to the fact that each node is located on a loop which surrounds the POI when

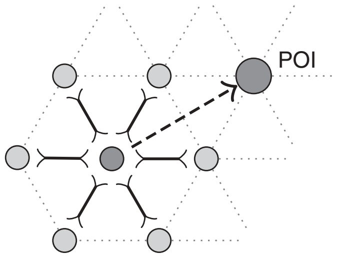


Figure 7.16 Neighboring repulsion and POI attraction.

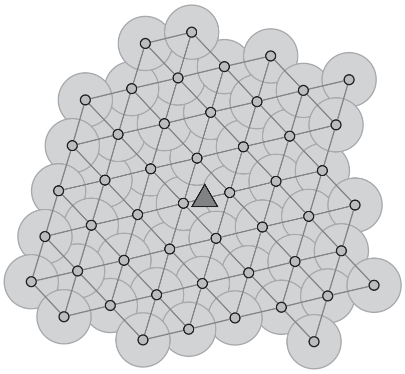


Figure 7.17 Topology resulting from 50 randomly deployed nodes.

the nodes come into equilibrium. Each node definitively stops moving when a certain number of changes in directions for obtuse angles, called *oscillations*, are detected. The final topology was proved to be biconnected if the network is stable, and the proposed protocol was shown to be highly scalable. Another advantage of this solution is that no communication is ever required beyond one hop.

REFERENCES

Abbas S, Mosbah M, Zemmari A. "Distributed computation of a spanning tree in a dynamic graph by mobile agents". IEEE International Conference on Engineering of Intelligent Systems (ICEIS 2006). Paphos, Cyprus; 2006. pp. 1–6.

Awerbuch B, Kutten S, Mansour Y, Patt-Shamir B, Varghese G. "Time optimal self-stabilizing synchronization". STOC'93: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing. New York: ACM; 1993. pp. 652–661.

Baala H, Flauzac O, Gaber J, Bui M, El-Ghazawi T. "A self-stabilizing distributed algorithm for spanning tree construction in wireless ad hoc networks". J Parallel Distrib Comput 2003; 63:97–104.

Basu P, Redi J. "Movement control algorithms for realization of fault-tolerant ad hoc robot networks". IEEE Netw 2004; 18(4):36–44.

Bettstetter C. "On the minimum node degree and connectivity of a wireless multihop network". MobiHoc'02: Proceedings of the 3rd ACM International Symposium on Mobile Ad hoc Networking & Computing. New York: ACM; 2002. pp. 80–91.

Blough DM, Leoncini M, Resta G, Santi P. "The k-neighbors approach to interference bounded and symmetric topology control in ad hoc networks". IEEE Trans Mobile Comput 2006; 5(9): 1267–1282.

Burman J, Kutten S. "Time optimal asynchronous self-stabilizing spanning tree". Distributed Computing, 21st International Symposium (DISC). Lemesos, Cyprus; 2007. pp. 92–107.

Carle J, Simplot-Ryl D. "Energy-efficient area monitoring for sensor networks". Computer 2004; 37(2):40–46.

Casteigts A, Chaumette S, Guinand F, Pigné Y. "Distributed maintenance of anytime available spanning trees in dynamic networks". Technical Report, RR-1457-09 LaBRI. University of Bordeaux; 2009.

Das S, Liu H, Nayak A, Stojmenovic I. "A localized algorithm for bi-connectivity of connected mobile robots". Telecomm Syst 2009; 40(3):129–140.

Falcon R, Nayak A, Stojmenovic I. "Reliability oriented robot augmentation protocol". Technical Report, University of Ottawa, 2009. In preparation.

Gaertner FC. "A survey of self-stabilizing spanning-tree construction algorithms". Technical Report, 2003.

Gallais A, Carle J, Simplot-Ryl D, Stojmenovic I. "Localized sensor area coverage with low communication overhead". PERCOM '06: Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications. Washington, DC, USA; 2006. pp. 10–337.

Guang T, Jarvis SA, Kermarrec A-M. "Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks". 2008. pp. 429–437.

Jorgic M, Goel N, Kalaichevan K, Nayak A, Stojmenovic I. "Localized detection of k-connectivity in wireless ad hoc, actuator and sensor networks". Proceedings of 16th International Conference on Computer Communications and Networks. Honolulu, Hawaii, USA; 2007. pp. 33–38.

Jorgic M, Stojmenovic I, Hauspie M, Simplot-Ryl D. "Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks". Proceedings of the Third Annual IFIP Mediterranean Ad Hoc Networking Workshop, MedHocNet. Bodrum, Turkey; 2004. pp. 360–371.

Li X. "Deploying sensors for optimal coverage by bi-connected mobile robot team". Technical Report, University of Ottawa, 2009. In preparation.

Li X, Frey H, Santoro N, Stojmenovic I. Localized self-deployment of mobile sensors for optimal focused-coverage formation. Technical Report, Ottawa: Carleton University; 2007.

Li N, Hou JC, Sha L. "Design and analysis of an MST-based topology control algorithm". Proceedings of INFOCOM. San Francisco, CA, USA; 2003.

Liu H, Chu X, Leung Y-W, Du R. "An efficient physical model for movement control towards bi-connectivity in robotic sensor networks". 2009. In preparation.

Melodia T, Pompili D, Gungor VC, Akyildiz IF. "A distributed coordination framework for wireless sensor and actor networks". MobiHoc'05: Proceedings of the 6th ACM International Symposium on Mobile Ad hoc Networking and Computing. New York: ACM; 2005. pp. 99–110.

Ovalle-Martinez FJ, Stojmenovic I, Garcia-Nocetti F, Solano-Gonzalez J. "Finding minimum transmission radii for preserving connectivity and constructing minimal spanning trees in ad hoc and sensor networks". J Parallel Distrib Comput 2005; 65(2):132–141.

Penrose MD. "On k-connectivity for a geometric random graph". Random Struct Algor 1999; 15(2):145–164.

Santi P, Blough DM. "The critical transmitting range for connectivity in sparse wireless Ad hoc networks". IEEE Trans Mobile Comput 2003; 2(1):25–39.

Tan HO, Korpeoglu I, Stojmenovic I. "A distributed and dynamic data gathering protocol for sensor networks". AINA'07: Proceedings of the 21st International Conference on Advanced Networking and Applications. Washington (DC): IEEE Computer Society; 2007. pp. 220–227.

Xu C, Li X, Nayak A, Stojmenovic I. "DHP: a delay-constrained power-efficient data aggregation scheme in sensor-actor networks". 2009. Submitted for publication.

Zou Y, Chakrabarty K. "Sensor deployment and target localization based on virtual forces". Volume 2; 2003. pp. 1293–1303.

Chapter 8

Location Service in Sensor and Mobile Actuator Networks

Xu Li, Amiya Nayak, and Ivan Stojmenovic
School of Information Technology and Engineering, University of Ottawa Ontario, Canada K1N 6N5

Abstract

In location service problem, mobile actuators send location update messages, whereas stationary sensors send search messages to learn latest position of actuators. The task is to minimize combined update and search message cost, while maximizing success rate of finding target actuator and subsequently routing to it. In the literature, many location service algorithms have been proposed for mobile ad hoc networks, and they can be directly applied to sensor and mobile actuator networks. This chapter reviews research efforts on this topic.

8.1 INTRODUCTION

In sensor and mobile actuator networks, actuators operate autonomously with no fixed infrastructure or centralized control. They determine their own location by the use of global positioning system (GPS) or some other type of positioning system, and register with location service. Location service tracks actuators' location and enables sensors to discover actuators so that geographic routing or other position-based algorithms can be applied. As an active subject, location service has been studied for over a decade in wireless ad hoc networks. Existing solutions can be directly applied to emerging sensor and actuator networks.

Location service has two ingredients: *location update* and *actuator search*. After an actuator leaves its current position, it needs to update its location in the network so that others can find it and keep routing packets to it. There exist two basic approaches for routing toward an actuator. In the first approach, when a sensor wants to route a packet by a geographic routing protocol such as Greedy-Face-Greedy (GFG) (Bose *et al*., 1999) to an actuator, it first searches for that actuator's latest location. In most cases, the search ends up at destination location, followed by report from destination back to the source, containing the exact position of destination. Since the position of the source can be included in the search message, this report can be carried by a georouting task. Alternatively, the source may use currently available and inaccurate information on the location of destination and route immediately toward that location, in the hope that the position information would become more accurate as the message approaches region containing destination.

As any other wireless ad hoc networking protocol, a location service algorithm is expected to be *efficient* in bandwidth and energy usage, *robust* against node mobility and node failure, and *scalable* to large-sized networks. Considering its unique goal, it is also required to have the following properties:

- *Discovery Guarantee:* It guarantees success of actuator search in arbitrary networks, as long as designated actuator is available.
- *Load Balancing:* It generates balanced load among network nodes with no storage or communication hotspot.
- *Locality Awareness:* It ensures successful actuator search within distance proportional to the distance from source to actuator.

In Section 8.2, we present a classification of existing location service algorithms. In Section 8.3, location update policies are discussed. Sections 8.4, 8.5, and 8.6 review some typical location services that cover a range of design choices.

8.2 CLASSIFICATION OF LOCATION SERVICES

Several classification methods have been proposed for existing location service algorithms in Stojmenovic (2002b), Camp *et al*. (2002), and Das *et al*. (2007). Here, in this chapter, we shall present a classification as a modification of Das *et al*. (2007). As depicted in Figure 8.1, at its top level are three classes, *flooding-based*, *quorum-based*, and *home-based*, each with two subclasses at the second level.

8.2.1 Flooding-Based Approach

This approach relies on flooding, which usually involves all or large portion of nodes in the network, for location update and actuator search. It can be further divided into two subclasses: *proactive* and *reactive*.

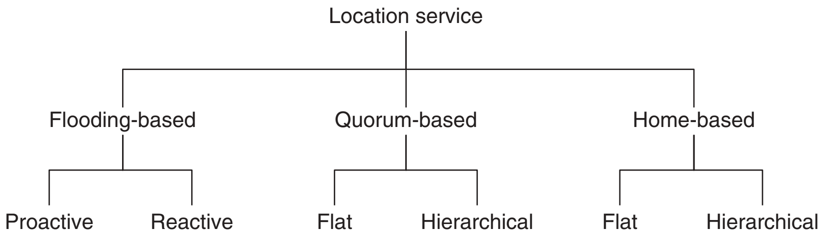


Figure 8.1 Classification of location services.

In proactive scheme, each actuator updates its location by flooding the network periodically or when desirable. If flooding is restricted only within some areas, actuator search will be directed to those areas, and it is sufficient that first receiver nodes in the areas respond. If the flooding area, however, covers entire network, then no actuator search is needed, as every node maintains the most recent locations of actuators simply by listening to their flooding messages.

In reactive scheme, if a sensor cannot find fresh location of a target actuator, it floods the network with a search message; currently recorded location and mobility information of the actuator can be used to narrow the scope of flooding. This approach normally does not have location update process.

8.2.2 Quorum-Based Approach

In this approach, location update and actuator search are directed to two *different* subsets of network nodes. The two subsets are respectively called *update quorum* and *search quorum*. They are carefully selected such that their intersection is not empty. As common (rendezvous) nodes of update and search quorums can provide the location information to the querying node as desired, success of actuator search is guaranteed.

Compared with flooding-based approach, quorum-based location service has less communication overhead as no potentially network-wide flooding is used. The challenge is how to select appropriate members for matching quorums to produce rendezvous with smallest cost. Depending on whether a recursively defined hierarchical structure is used for quorum formation, a quorum-based location service can be further classified as *flat* or *hierarchical*.

8.2.3 Home-Based Approach

In this approach, each actuator selects a home region that is known to others, and proactively sends location updates to nodes located in or closest to that region. In order to locate target actuator, sensors send search messages toward home region of the actuator, and the messages may possibly be redirected from there to current location of the actuator.

Home region could be a geographic area or point, determined by actuator initial location or a hash function of actuator ID. Multiple home regions may be used to reduce search cost, increase locality awareness, and improve robustness, at the cost of location update messages. This approach can be viewed as a special case of quorum-based approach, where update quorum and search quorum are the same, and similarly divided into two subclasses: *flat* and *hierarchical*.

8.3 LOCATION UPDATE POLICIES

Almost all types of location service involve location update. The purpose of location update is to propagate latest information of actuator location. There are different location update policies, for example, *distance-based*, *movement-based*, *time-based*, and *connectivity-based*, which may result in different performance in message cost.

In distance-based update, an actuator updates its locations whenever its distance to last reported position is long enough, for example, beyond a threshold value. Considering *distance effect* (Basagni *et al.*, 1998) (i.e., the larger the distance between two nodes, the slower they appear to move with respect to each other), the actuator need update locations to nearby sensors more frequently than to those farther away. This can be implemented by associating with each location update message an "age" indicating how far from the message can travel its sender, as suggested in Basagni *et al.* (1998).

In movement-based update, an actuator updates its location whenever it completes a predefined total "mileage" for mobility since last location update; in time-based update, it updates its location at some time intervals. In connectivity-based update, an actuator updates its location when its local network topology changes to some extent since the last update; such change can be decided according to geographic position information, as proposed in Stojmenovic *et al.* (2000) and Su *et al.* (2000). Topology changes can be discovered from changes in neighborhood in periodic beacons heard from neighbors. Changes can even be predicted using estimated speeds and directions of movement of nodes.

Karumanchi *et al.* (1999) discussed when to update. They argued that distance- and movement-based updates have limited usefulness in ad hoc network and can induce unnecessary messages, for example, when nodes move jointly in the same direction, or move within a small circle. They experimentally concluded that the best strategy is connectivity-based update, whenever a certain prespecified number of incident links have been established or broken since the last update.

8.4 FLOODING-BASED ALGORITHMS

In this section, five representative flooding-based location service algorithms have been reviewed: doubling circle update (Amouris *et al.*, 1999), direction-based update (Friedman and Korland, 2005), localized update (Yang *et al.*, 2009), request zone search (Stojmenovic *et al.*, 2003, 2006), and expanding ring search (Kasemann *et al.*, 2002). The first three schemes belong to proactive category, while the last two belong to reactive category.

8.4.1 Doubling Circle Update

Amouris *et al.* (1999) presented a doubling circle location update scheme. In this scheme, each actuator propagates its location information within circles $C(i)$ of increasing radii $2^i R$ for $i = 1, 2, 3, \ldots$ Each of these circles is associated with a refreshment timer. Whenever the timer expires (time-based policy), the actuator broadcasts a location update message within the corresponding circle. In addition, whenever the actuator moves outside a circle $C(t)$ for some t (distance-based policy), it broadcasts its location to all the nodes located within a circle of radius $2^{t+1}R$ centered at its current position.

Actuator search (or direct routing to it) then follows these circles of last updates. Sensors (source or intermediate sensors) forward a search message toward the last reported position of target actuator, which since the last report may have moved within the circle of some radius. As the message moves closer to target actuator, its position information becomes more precise, and sensors are able to direct the message toward center of circles with twice smaller radius than previously, until target actuator is eventually reached. For instance, as illustrated in Fig. 8.2, sensor S sends a message toward the last known position D of a target actuator; on its way, the message is redirected to more recent position D' and finally to exact position D''.

Doubling circle update scheme is proactive flooding-based location service. As a large update circle may contain all the network nodes, location update may possibly convert to flooding, leading to large amount of message overhead and limited scalability. A similar algorithm, using squares rather than circles, and additional sophisticated techniques, is proposed in Li *et al.* (2000).

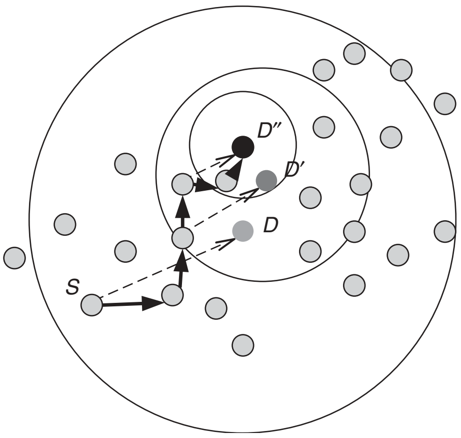


Figure 8.2 Doubling circle search and update.

8.4.2 Direction-Based Update

Friedman and Korland (2005) presented a direction-based location update scheme. The network area is divided into a two-dimensional grid. Each actuator floods entire network with its location at initiation. Then, each sensor maintains the relative direction (*Right*, *Up*, *Left*, or *Down*) for routing a message to every actuator along the grid. Location update obeys distance-based policy. Two variants LS1 and LS2 were proposed.

Consider grid cell T in which an actuator is located. In LS1, all cells for which T is on their right side and not above them are marked *Right* (the mark is stored by nodes located in those cells); all those for which T is above and not to their left are marked *Up*; and so on, as shown in Figure 8.3(a). In LS2, all cells for which T is on their right side and below them are assigned two marks *Right* and *Down*; all those for which T is on their right and above them are assigned *Left* and *Up*; and so on, as shown in Figure 8.3(c).

For ease of description, define array $DS = \{Left, Down, Right, Up\}$. Assume an actuator moves from its current cell to a neighboring cell in direction $DS[i]$. In LS1, the actuator updates after its movement the cells in direction $DS[(i+1)\text{mod}4]$ along part of its old residing cell array orthogonal to direction $DS[i]$, and those in direction $DS[(i+3)\text{mod}4]$ along part of its new residing cell array orthogonal to direction $DS[i]$, as shown in Figure 8.3b. In

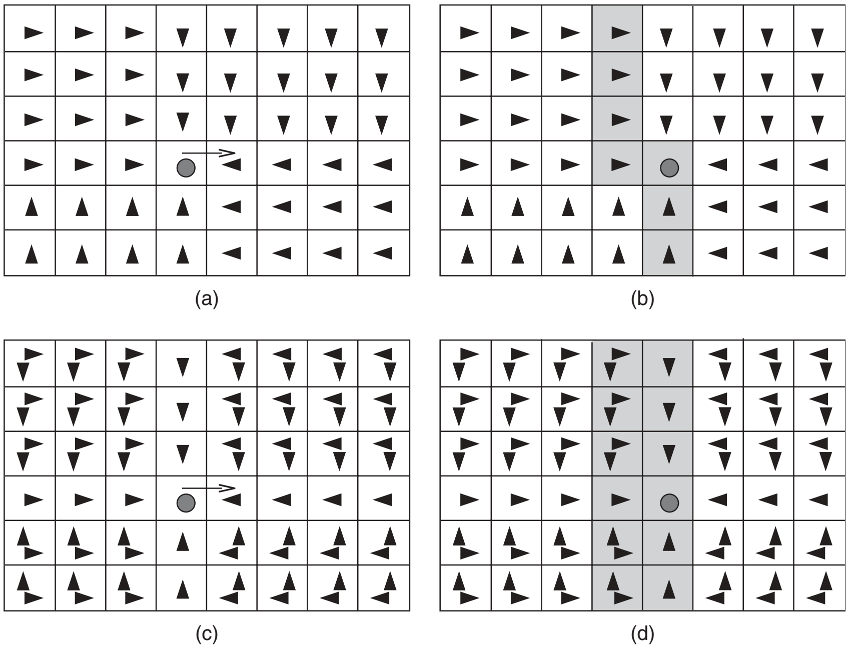


Figure 8.3 Direction-based update: (a) LS1—before movement; (b) LS1—after movement; (c) LS1—before movement; (d) LS2—after movement.

LS2, it updates its entire old residing cell array and its entire new residing cell array, as shown in Figure 8.3d.

To find a target actuator, a sensor sends a search message simply following those locally recorded relative directions of the actuator in grid cells. After receiving the search message, the actuator replies the sensor with its current location. This algorithm restricts location (precisely speaking, direction) updates within bounded areas and thus has reduced message overhead. However, the presence of empty cells requires modifications to the algorithm similar to those discussed here for the quorum-based approach.

8.4.3 Geographic-Routing-Based Update

Yang *et al.* (2009) presented a localized location update scheme for routing to a mobile actuator (e.g., data sink). The objective is to enable every sensor node to maintain a routing next hop to the actuator for data dissemination. Routing next hop is determined by geographic routing protocol GFG (Bose *et al.*, 1999). In this scheme, the actuator has the same transmission radius r_c as sensors. It broadcasts its location to every node in the network at initiation (once); then, it starts to move around in the network and periodically exchanges hello messages with neighboring sensors. When necessary, the actuator sends location update messages, which are selectively forwarded by receiver nodes. Both controllable mobility and uncontrollable mobility are considered. With controllable mobility, the actuator knows where it goes and at what speed; with uncontrollable mobility, it has no such knowledge. Two versions of the scheme were presented for the two mobility models, respectively.

In the version for uncontrollable mobility, the actuator monitors the radio connection to its neighbors by listening to their periodical hello messages. It considers a neighboring node *lost neighbor* if the node is being removed from its two-hop neighborhood, or *semi-lost neighbor* if the node is being removed from its one-hop neighborhood but remains within its two-hop neighborhood. A current neighbor that covers a semi-lost neighbor is called *recovery neighbor*. The actuator is able to classify neighbors because it collects their position.

Whenever a link breakage or a link creation occurs, the actuator sends a *long* location update message to its neighboring sensors. The message contains the actuator's latest location and a recovery neighbor list (which could be empty). The recovery neighbor list does not necessarily contain all the recovery neighbors; it is sufficient as long as all semi-lost neighbors are covered by listed nodes. Meanwhile, the actuator sends to lost neighbors (if any) a *short* location update message, which carries an empty recovery neighbor list, by routing protocol GFG. Once receiving a location update message, a sensor checks if its next hop to the new actuator position is different from that to the old one, and it also checks if it itself is among the recovery neighbor list. If either of the two answers is positive, it transmits by locally broadcasting the location update message (once); otherwise, it does not. A node always forwards a short location update message to the next hop toward the destination.

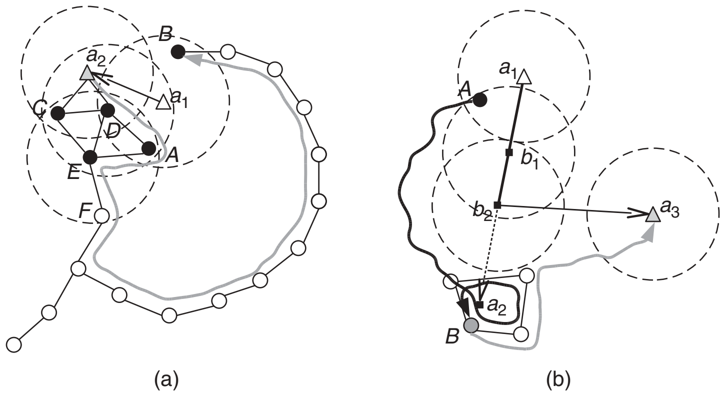


Figure 8.4 Geographic-routing-based update: (a) uncontrollable mobility and (b) controllable mobility.

Figure 8.4a illustrates how this scheme works with uncontrollable actuator mobility. The actuator moves from a_1 to a_2. At a_2, it recognizes lost neighbor B and semi-lost neighbor A. Then it sends a long location update message to its current neighbors C and D, and a short location update message to B along the path indicated by arrowed line. C retransmits the location update message because its next hop changes; D retransmits because it is a recovery neighbor (covering the actuator's semi-lost neighbor A). Node E receives the location update message from D, and retransmits the message because its next hop changes. Node A receives the update message from D and decides to retransmit it. Node F receives the message forwarded by E, but it does not retransmit the message because its next hop to the sink remains unchanged. The propagation of the long location update message finally stops at F. In the above process, each node receives and drops duplicated location update messages. While the short location update message travels along the path, it may be retransmitted by each intermediate node according to the same policy (those retransmissions are not shown in the figure).

In the version for controlled mobility, the actuator knows which (and when) of its incident links will be broken. Before link breakage, it sends the corresponding neighbors a location update message informing them its destination and moving speed. It sends such a location update message also to newly discovered neighbors. When a sensor receives the location update message for the first time, it checks whether its current next hop to the actuator and the next hop to the actuator destination are the same. If they are not identical, it retransmits the message. It then also estimates the actuator's arrival time and buffers data packets locally before delivery to the actuator's new position.

Data dissemination delay occurs due to local data buffering. To reduce the delay (and also the requirement for local storage space), the actuator is suggested to break a long trip into short segments and update its location for those intermediate destinations. During the course of relocation, the actuator may suddenly decide to move to another location possibly at a different speed. In this case, it sends its moving speed and the new destination toward the old destination using GFG. The node closest to the actuator's old location is guaranteed to receive the information. This node is called *anchor node*. It will later receive from sensors data routed to this old actuator destination and then redirect the data to the actuator's new destination. Anchor nodes form a routing backbone and ensure data delivery in the case of frequent unexpected actuator destination change.

Figure 8.4b illustrates how this scheme works with controllable actuator mobility. The actuator first moves from location a_1 toward location a_2. At location b_1 (where direct connection to node A would be lost), it sends to A its destination a_2 and moving speed. At location b_2, it changes its destination to a_3 and routes this change to anchor node B that is closest to the old destination a_2. Later, sensor A sends data toward a_2; the data is received by B and redirected to a_3.

Geographic-routing-based update scheme is proactive flooding-based location service. Its location update relies on flooding restricted only within necessary area, where nodes experience changes in routing to the actuator. Flooding area is not determined by the actuator but defined distributively by nodes' local decision on retransmission. This scheme is a localized approach. We believe it is a promising solution that leads to both message efficiency and scalability.

8.4.4 Request Zone Search

Ko and Vaidya (1998) and Basagni *et al.* (1998) independently described a request zone search scheme. Source or any intermediate sensor computes, according to the last reported location and mobility information of target actuator, a circular *expected zone* that covers the potential current locations of target actuator. Then, it sends a search message to all neighbors in an angular *request zone* determined by its tangents to the expected zone. But, as the request zone may not contain any node toward target actuator, it is possible that actuator search fails frequently especially in a network with a lot of void areas. In case of failure, the algorithm triggers network-wide flooding to recover, which may, however, induce increased message overhead.

LOTAR (Wu and Harms, 2000) and GRID (Liao *et al.*, 2001) are two simple variants of this request zone search scheme. In Wu and Harms (2000), more accurate expect zone could be calculated as any two neighboring sensors periodically exchange their location tables (containing location of all sensors in the network). In Liao *et al.* (2001), grid-based coordination instead of geographic coordination is used for routing.

Stojmenovic *et al.* (2003, 2006) modified the definition of request zone (Ko and Vaidya, 1998, Basagni *et al.*, 1998) to provide uniform framework with the corresponding notions in GEDIR (Stojmenovic and Lin, 2001) and MFR (Takagi and Kleinrock, 1984) routing methods. They presented V-GEDIR and CH-MFR methods, in which actuator search message is forwarded to exactly those neighbors that may be best choices for a possible position of target actuator. The request zone may include several neighbors that are outside the angular range, because they have the closest direction for the tangents to the expected zone. In V-GEDIR, these neighbors are determined by intersecting the Voronoi diagram of neighbors with the expected zone of target actuator, while the portion of the convex hull of neighboring sensors is analogously used in CH-MFR. Experiments show that these algorithms have higher success rate, and lower hop count and flooding rate than that of Ko and Vaidya (1998) and Basagni *et al.* (1998).

Observe Figure 8.5, where shaded area represents the transmission range of sensor S, and thick circle indicates the expected zone of actuator D. The neighbors A, B, C, K, and L of S are closer to the last known position of D than S itself. In V-GEDIR, the Voronoi diagram (marked by thick dashed lines) of these neighbors is constructed locally by S. Consider a bisector, for example, the one for B and C. S may determine that the expected zone of D is completely on one side of the bisector, and thus C is closer than B for any possible location of D. B is out of consideration, and forwarding sensors for S are therefore A and C. In CH-MFR, the convex hull (marked by thick solid lines) of these neighbors is used. Find the two neighbors whose projections on tangent lines from S are closest to the common points of these tangent lines with the request zone. Forwarding sensors for S are all neighbors that are located on the convex hull between these two neighbors (inclusive). In Figure 8.5, they are A and C.

8.4.5 Expanding Ring Search

Kasemann *et al.* (2002) presented a reactive location service (RLS) that employs an expanding ring search scheme for actuators. A similar location service that has the same name and can be viewed as a subset of RLS (Kasemann *et al.*, 2002) was independently suggested in Camp *et al.* (2002).

In RLS (Kasemann *et al.*, 2002), source floods in rounds a region of increasing radius d (in hop count), until target actuator is found or the maximum value of d is reached. The increment of d may be linear, exponential, or binary. Every

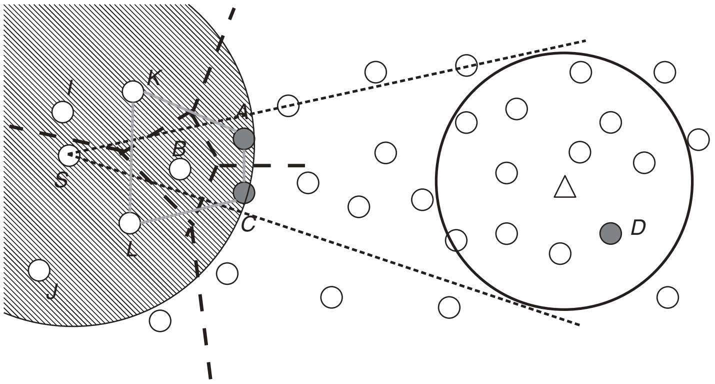


Figure 8.5 Formalized request zone search.

sensor monitors network traffic, and retrieves and catches embedded location information about actuators. Under this circumstance, actuator search may be answered early by some sensor before reaching the target actuator. Location catching reduces both search delay and message overhead (as subsequent flooding is no longer needed once a reply is received). Backoff time is used for each involved rebroadcast operation for the purpose of congestion control and fast expansion of flooding. It is determined in such a way that the farther away a sensor is from last message forwarder, the sooner it will rebroadcast the message. A combined distance-/counter-based rebroadcast suppression scheme is proposed to solve broadcast storm problem (Ni *et al*., 1999).

8.5 QUORUM-BASED ALGORITHMS

Quorum-based location service has been widely used in fixed networks and cellular networks (Prakash and Singhal, 1996; Prakash *et al*., 1997; Krishnamurthy *et al*., 1998). Its adaptation for wireless ad hoc networks is suggested in Stojmenovic (1999), Liu *et al*. (2006), and Stojmenovic *et al*. (2008). So far, several different variants (e.g., Abraham *et al*., 2004; Li *et al*., 2008; Liu *et al*., 2009) of the quorum technique have been proposed for location service. In this section, we review these existing work in detail.

8.5.1 Strip Quorum

Stojmenovic (1999), Liu *et al*. (2006), and Stojmenovic *et al*. (2008) proposed a localized strip quorum technique for wireless ad hoc networks with connectivity-based location update policy. Each actuator monitors the state of its incident links. Whenever a link breakage/creation occurs (possibly due to its own movement), it reports its current position to its neighbors. After a certain number of link changes, it forwards its current position to all the nodes located in a "column." That is, it sends its location in both north and south direction to reach the north and south boundaries through GFG routing protocol (Bose *et al*., 1999). The nodes along this column form an update quorum. Note that GFG will route the location update message along the outer boundary of the network, which is thus included in the update quorum.

A source sensor queries its q-hop neighborhood for target actuator's location. If the answer is negative, or if its obtained information is not fresh enough, the search continues in the east and the west direction. These searches are performed independently. The traces of the eastbound search and the westbound search form a "row", that is, a search quorum, which intersects the update quorum of every actuator.

As the search message travels along the search quorum, it picks the latest location information about target actuator. After reaching the end (westmost node or eastmost node) of the quorum, it is forwarded to target actuator, which then replies source directly with correct location. Alternatively, the intersection sensors of the search quorum of source and the update quorum of target actuator may reply immediately if their stored information about target actuator is sufficiently fresh.

Figure 8.6 illustrates a variant, called CR+CR in Stojmenovic (1999), Liu *et al.* (2006), and Stojmenovic *et al.* (2008), of the strip quorum technique. In this variant, actuator D sends location updates in four geographic directions to form two update quorums. Sensor S sends search messages for D in four directions to construct two search quorums. The two search quorums intersect the two update quorums; nodes at the intersection points can provide S with D's location. Note that this figure shows update quorums of only actuator D. In the case of multiple actuators, every actuator will have similar update quorums as D, and all update quorums together will form a mesh structure.

This quorum technique has obvious advantages. Every sensor can discover every actuator without network-wide querying. Because location update and actuator search are restricted within two strips, that is, a column and a row, communication overhead is greatly reduced. No particular node is designated to store a certain actuator's location, and thus no bottleneck is created in the network. If target actuator is nearby, source will get answers quickly since update quorum and search quorum intersect earlier than in the case that target actuator is far apart. However, this scheme also has nonnegligible weaknesses. Location update and actuator search have to cross the entire network; to guarantee row–column intersection, network outer boundary has to be included into every quorum, adding large storage load on boundary nodes and making their battery power drain out fast.

Considering the fact that sensors are static, Yu *et al.* (2009) proposed to avoid overloading all boundary nodes by sacrificing only extreme sensors in the four directions. The idea is to replace update-/search-triggered outer boundary traversal with a preprocessing step. In this step, boundary nodes are detected by an

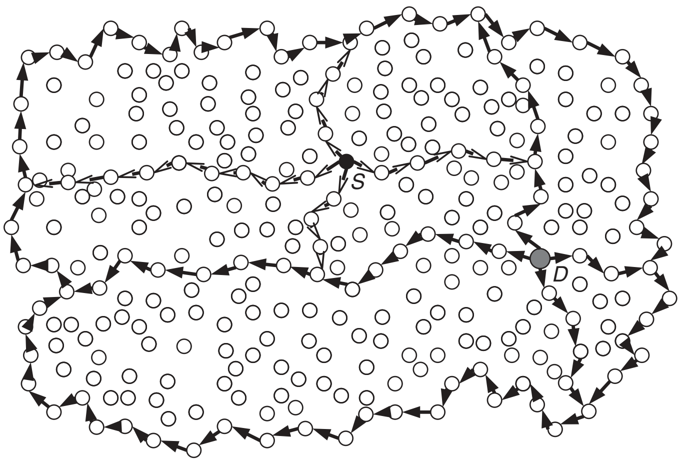


Figure 8.6 Strip quorum (CR+CR variant).

existing boundary detection algorithm; then, extreme nodes in all four directions in the whole network are identified by face traversal starting from a predefined initiator node. After this step, location updates are directed to northernmost and southernmost nodes, while searches are routed to extreme nodes in west and east directions. These two routes (quorums) are guaranteed to intersect; the sensor at intersection answering the location query.

In dense networks, the strip quorum scheme has degraded performance because too many nodes become involved in processing search messages. This problem is addressed in Melamed *et al.* (2007), and Zhang *et al.* (2007) by proposing the network division into equal size grids, and selecting one leader sensor in each grid to construct a backbone. Backbone sensors are connected through other sensors, and strip quorum then continues mainly on the backbone nodes. The advantages and disadvantages of grid-based quorum are discussed in Liu *et al.* (2007), where the authors also introduced a superior improvement based on localized connected dominating set (CDS) by restricting location update and actuator search only among sensors from CDS.

8.5.2 Mesh Quorum

Li *et al.* (2008, 2009) presented a localized mesh quorum technique, called *iMesh*, for nearby actuator discovery. The term *nearby* implies that discovered actuator is at most twice as far as closest one. This quorum scheme is based on a novel planar structure *information mesh* created by the use of a formalized blocking rule (Tchakarov and Vaidya, 2004) in quorum-based mesh construction.

In iMesh, actuators send location update messages to four geographic directions, that is, north, west, south, and east, by routing protocol GFG (Bose *et al.*, 1999). During their propagation, these messages collinearly or orthogonally block each other according to a local blocking rule: node receiving location update messages from multiple actuators forwards only the message of the closest one.

In the presence of asynchrony, the blocking rule may be violated. But, nevertheless, wrong transmission can be locally identified and fixed by nodes at which the blocking rule is supposed to be applied. These nodes send revocation message following the forward path of those wrongly forwarded location updates to erase inconsistent information. The proper propagation paths of location updates form an *information mesh*, as shown in Fig. 8.7. Note that the outer boundary of the network is included in the mesh structure according to the property of GFG (Bose *et al.*, 1999).

To discover the location of a nearby actuator, source S simply conducts a cross lookup process within its residing mesh cells. That is, they send actuator search messages in four geographic directions and the messages are guaranteed to reach the premier of its home mesh cell, as illustrated in Figure 8.7. Then, the rendezvous nodes reply to the source with the information of its recorded closest actuator.

The blocking rule reduces the possibility of an actuator being discovered, while restricting message transmission and reducing communication overhead.

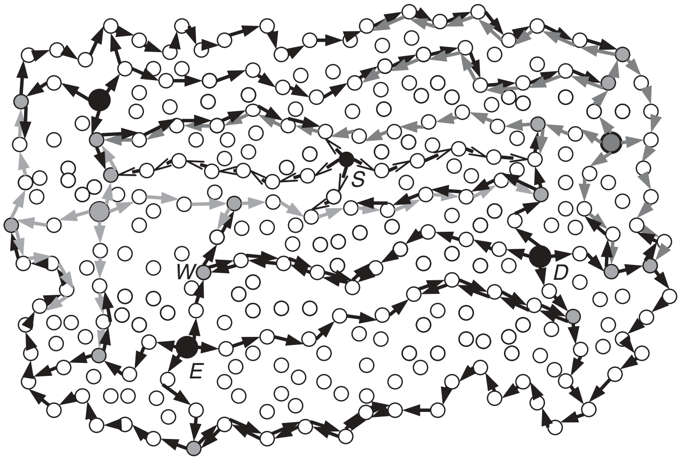


Figure 8.7 Mesh quorum.

Li *et al*. (2008, 2009) characterized the cases where nearby/closest actuator selection is violated, and proposed an *extension rule*: a node W, where information from E orthogonally blocks information from D, transmits E's information along the backward transmission path of D's information for a limited distance, as shown in Figure 8.7. The extension rule does not change the structure of information mesh. But, it effectively decreases the occurrence possibility of undesired remote actuator selection.

Through analytical study, the authors show that iMesh has significantly lower message complexity than strip quorum method (Stojmenovic, 1999; Liu *et al*., 2006; Stojmenovic *et al*., 2008) and that it generates constant per node storage load, which is a unique property that no other quorum-like algorithm possesses. Extensive simulation shows that iMesh guarantees nearby (closest) service selection with probability $>99\%$ (resp. 95%).

The authors indicated how to deal with node mobility: before an actuator starts to move, it initiates a revocation process to remove its own information from information mesh, and after it becomes stabilized, it acts as a newcomer to update its location.

8.5.3 Hierarchical Spiral Quorum

Abraham *et al*. (2004) presented a locality-aware location service using hierarchical spiral quorums. Each actuator is hashed to a point in the network area according to its ID. For each actuator, the network area is recursively partitioned, with its hash point as origin (note that this is the only use of nodal hash point in the algorithm), into a square hierarchy. At the lowest level (level 0), each square has predefined size; four neighboring level-k ($k \geq 0$) squares form a square at level $k + 1$; the highest level square covers entire network area. This square hierarchy is locally computable to each actuator.

Consider an arbitrary actuator A. It has a residence square, the one where it is located, at each level in its own square hierarchy. The corner points of all these residence squares form a virtual spiral that surrounds A and exponentially increases in distance, as shown in Fig. 8.8 where H is the hash point of A. When necessary, A updates its location along this spiral. A sensor computes the square hierarchy of actuator A using actuator ID, and performs actuator search along a similar spiral around itself in the hierarchy. The two spirals must intersect because the points in both spirals are computed using the same hierarchical squares; the location of A is found at the intersection points that do not necessarily include A's hash point H. Spiral-like message transmission is supported by a combination of geographic routing and iterated bounded flooding.

At each level of its square hierarchy, an actuator publishes location not only to the four corners of its residence square but also to the corners of the eight surrounding squares. Location update is then performed only when the actuator moves out of the nine squares boundary. To reduce cost, actuator location is stored only at level 0; at level $k > 0$ stored pointers point to the level-$k - 1$ squares where actuator may be located. This enables lazy updates; that is, an actuator does not update its level-k location pointers unless it moves a total of a certain distance proportional to 2^{k-1}.

Because actuators have different hash points, their square hierarchies and thus location update spirals are different. This difference leads to balanced store load among nodes. It is proven that actuator search has desired message complexity

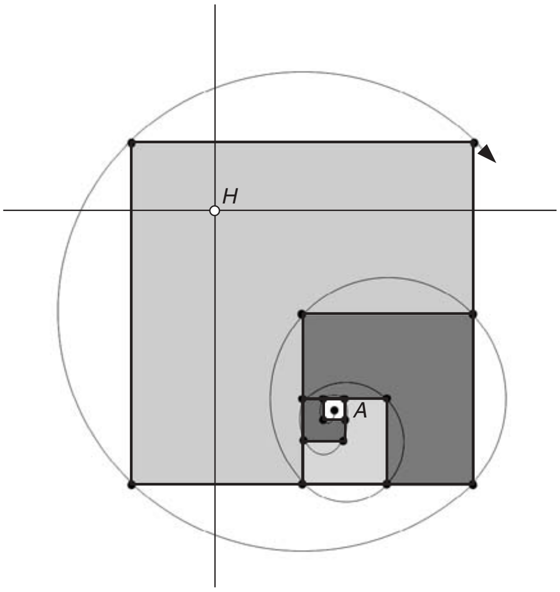


Figure 8.8 Hierarchical spiral quorum.

$O(d)$ in the average case, but undesired large message complexity $O(d^2)$ in the worst case. Here d is the minimal path length between source sensor and target actuator. If an actuator moves distance d, the average cost of location update is $O(d \log d)$.

8.5.4 Hierarchical Ring Quorum

Liu *et al.* (2008) proposed a hierarchical ring quorum scheme. In this scheme, every actuator constructs and maintains a hierarchy of rings for itself. These rings have doubly increased radii, that is, the radius of order-k ring is twice as large as order-$(k-1)$ ring; they are all centered at the actuator at initiation, and may no long have a common center as the actuator moves.

Order-0 ring is also called *core ring*. Each actuator reconstructs its core ring with its current location as it moves (so that the ring is always centered at its immediate location), and proactively updates location to all sensors in the new ring by flooding. When an actuator leaves its order-k ($k > 0$) ring, it reconstructs the ring with its current location as center, and sends location update to sensors only along the new ring. Location update is time-stamped so that location freshness can be measured. Figure 8.9 shows how an actuator D performs location update. The actuator moves along the trajectory $P_0, P_1, \ldots, P_5$. Broken circles are the rings it constructed in the course of its movement; its current hierarchical rings $R_D^0, R_D^1, \ldots, R_D^3$ are shown by continuous circles.

Sensors also maintain hierarchical rings for themselves. These rings are used for actuator search. Actuator search consists of two phases. In the first phase, source sensor S firsts searches location of target actuator D along its core ring; if the search is not successful, it will search along order-1 ring, and so on. The

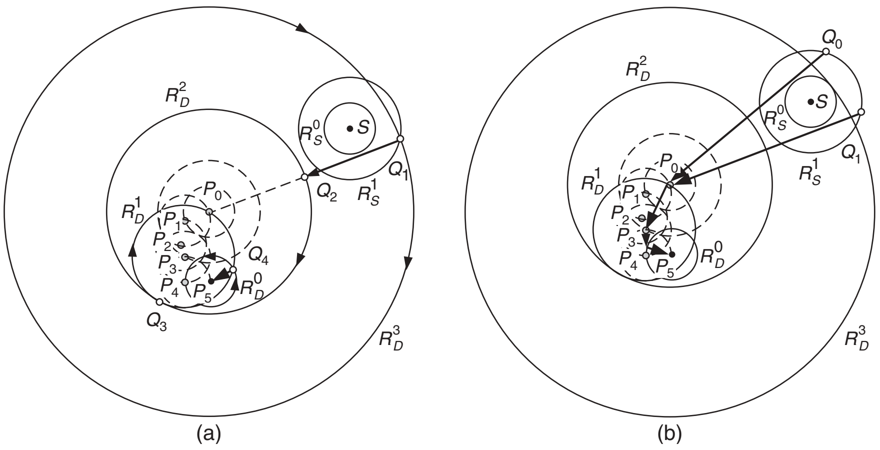


Figure 8.9 Hierarchical ring quorum: (a) perimeter-based search and (b) direct search.

search goes up higher in the ring hierarchy and eventually reaches a ring that is large enough to intersect a ring of D. Then, the second phase begins.

The second phase can be carried out by two different ways, which are here referred to as *perimeter-based* method and *direct* method, respectively. In perimeter-based method, actuator search switches between rings of D at their intersection points and always follows a fresher ring in clockwise direction. If the search makes a full circle on certain ring, then it is directed to the best known location of D. On its way, perimeter search resumes when yet fresher ring of D is found. Finally, the search reaches D, and D replies S with its latest location. This method suffers from long spiral-like search paths.

In direct method, actuator search is always directed to the last reported location of D. As the search message gets closer to D, it hits all the lower-order rings of D, and is redirected toward more and more accurate location of D. When it enters the core ring of D, it is redirected by the first receiver sensor to D, which then replies S with its latest location. The two actuator search methods are illustrated in Figure 8.9, where actuator D is located at position P_5 and search paths are highlighted and marked by arrows. Obviously, direct method has better locality awareness than perimeter-based method.

If source sensor and target actuator are geographically close to each other, their ring hierarchy will intersect at a low level, and, therefore, actuator search will be directed to target actuator soon without expanding far from source. Large ring at high level of the hierarchy will be traversed only when source and target actuator are far apart from each other. This hierarchical algorithm achieves good locality awareness while avoiding large-extent flooding, and thus has less overall communication cost.

8.6 HOME-BASED APPROACHES

Stojmenovic (2002a), and Blazevic *et al.* (2001), and Woo and Singh (2001) independently suggested flat home-based location service approaches that are similar to Mobile IP and cellular phone network. Hierarchical home-based location service was first proposed by Li *et al.* (2000). A number of home-based location service, for example, Xue *et al.* (2001), Viana *et al.* (2005), Chen *et al.* (2006), and Kieb *et al.* (2004), have been proposed so far, by adopting different home region definition methods. They show close analogy to the above pioneer work in design. In this case, below we will cover only the representative Stojmenovic (2002a) and Li *et al.* (2000) in detail.

8.6.1 Flat Home Region

Stojmenovic (2002a) presented a home-agent-based location service. In this algorithm, an actuator's home agent is a circular area of radius R centered at the actuator's initial position. Here, R is a predefined value proportional to the actuator communication radius. The location of home agent is flooded to the network.

Alternatively, hash function is used to find location of home region (Blazevic *et al.*, 2001; Woo and Singh, 2001).

Each actuator sends location update messages toward the center of its home agent by a geographic routing such as GFG (Bose *et al.*, 1999). Once an update message enters its originator's home agent base, it is forwarded in a strictly greedy manner. If greedy next hop cannot be found, current sensor will transmit the message to all the nodes within radius R, by flooding, or intelligent broadcasting (Stojmenovic *et al.*, 2002), or using larger transmission radius (if applicable). Each actuator, when transmitting anything, also uses the opportunity to broadcast its own new location. Sensors monitor network traffic and cache-embedded location information about actuators.

Sensor S issues two search messages for target actuator D. One is sent to D using currently known location of D by geographic routing. On the way, it is updated with more recent location of D by intermediate nodes. The other message is sent to the home agent of D in the same way as a location update message. Let W be the sensor (if any) in D's home agent that stops the search message because it is the local minimum. W will send a request to all the sensors within a circle, centered at itself, of radius R. This request message contains the most recent location information collected on the way of the search message to W. All sensors inside the circle with yet more recent location information will reply. Then, W uses the freshest location of D obtained from reply messages to redirect the search message to D. Upon receiving the search message for the first time, destination D replies S with its exact location.

Figure 8.10 illustrates this home-agent-based location service. The original location of destination d is at D_1, and its home agent base is the circular area

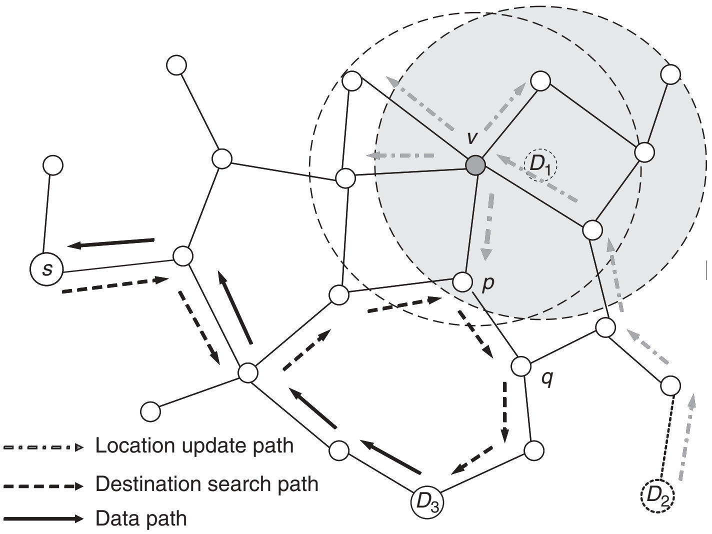


Figure 8.10 Home-agent-based location service.

centered at D_1. Later, d moves to D_2 and sends location update back to its home agent from there. Node v is the node closest to D_1. After it receives d's location update message, it broadcasts the message within the circular area centered at itself. Source S sends a destination search message to the home agent based on d. Node p receives this message and redirects the message toward D_3. When node q receives the message, it finds that destination d already moved to location D_3, it then redirects the message to D_3. After d gets the message, it replies S with its current location and meanwhile builds a route.

This algorithm has weakness in locality awareness. Even if target actuator is currently close to source, source may still have to access the remote home agent of target actuator for its location, thus increasing both message overhead and search delay. In addition, success of actuator search relies on the occupancy of home agent. Search fails when home agent nodes all move out of the home agent base. In Blazevic *et al.* 2001, it is suggested to vary home region radius so as to keep an approximately constant number of internal nodes. But, this method requires centralized computation for node density inside home region.

In a recent variant (Viana *et al.*, 2005), Hilbert curve is used to define locally expandable home region and thus achieve increased success rate of actuator search. Every node is associated with a vertex in a Hilbert curve filling the network area, and assigned a *control region*, which is the region filled by the segment between the node's predecessor and successor on the Hilbert curve. Each node maps its ID to a Hilbert vertex by a public hash function, and takes the corresponding Hilbert region as home region. When the only occupant of a subsquare leaves, its control region will be taken over by one of its predecessors and its successors that have a smaller control region.

In Xue *et al.* (2001), multiple home regions are used to improve locality awareness and tolerate empty home regions; different home regions may contain location information of different levels of accuracy, and only a small set of home regions need to be updated when the node moves. However, no matter how many home regions are used, movement of all nodes makes all home regions empty and unreachable. This is in fact the inherent weakness of home-based location service (whether flat or hierarchical). In this aspect, quorum-based approach is more flexible and advantageous as it uses dynamically selected network nodes for location update and actuator search.

8.6.2 Hierarchical Home Region

Li *et al.* (2000) presented a grid location service (GLS). The network area is evenly partitioned into a number of order-1 squares. Four neighboring order-k ($k \geq 1$) squares form an order-$(k+1)$ square; an order-k square is part of only one order-$(k+1)$ square. By this means, a quad-tree is established over the squares of different sizes, and, in the hierarchy, each node is located in exactly one square of each size. This partition serves as the base of GLS and is global knowledge. Actuators and sensors are assigned identifiers in a uniform and distinct way. Location update obeys distance-based policy.

A node (whether sensor or actuator) T selects as *location server* three nodes for each level of the hierarchy, one from each of its adjacent three squares. Specifically, it sends a location update message to a selected order-k square using geographic forwarding. The first receiver node W in the square starts an update process: it forwards within the square the message to a node whose ID is closest to (i.e., the least greater in a circular ID space than) T's, which in turn forwards the message in the same way. The process terminates when the message reaches a node square-wide closest to T in terms of ID, and this node becomes location server of T in the square.

The reason this update process works is that the nodes in an order-1 square are required to immediately exchange their location information at start-up, and that all the nodes in order-k square have distributed their location throughout the square in previous processes intended for lower-order squares.

Destination (whether sensor or actuator) search is performed in a similar way as location update. Source node S sends a search message, carrying its current location, to a node whose ID is closest to destination D, for which it currently has location information. Each intermediate node forwards the message in the same way. Eventually, the message will reach a location server of D. By geographic forwarding, this location server forward the message to D, which then directly replies S with its current location. To tolerate mobility, before a node moves to another order-1 square, it leaves a "forwarding pointer" in its current order-1 square. This pointer can be used to locate that node if search intended for the node arrives.

Figure 8.11, where node IDs are presented and used to denote nodes, illustrates GLS. In this example, node B, whose ID is 17, chooses its location servers,

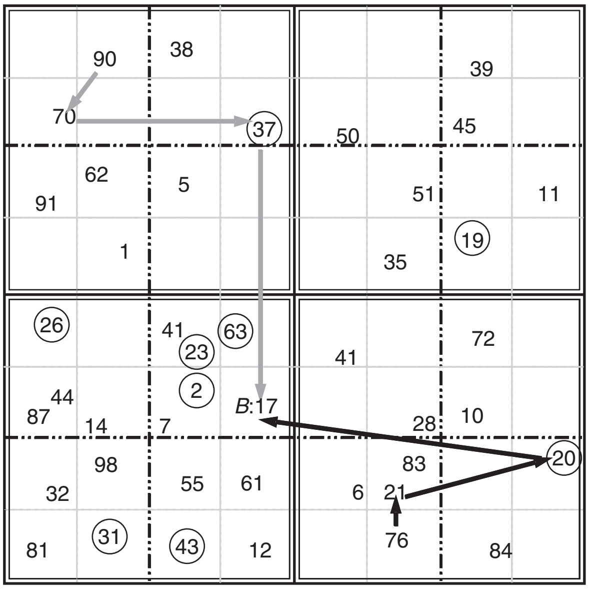


Figure 8.11 Grid location service.

whose IDs are circled, in the grid hierarchy; two destination searches for node *B* are originated, respectively, from nodes 76 and 90, and their search trails are highlighted by arrows.

Grid location service has good scalability because of its hierarchical design. A node's location servers are relatively dense near the node, and sparse farther away from the node. This ensures improved locality awareness as sources near destination can use a nearby location server to find the location of destination. Grid location service may cause large message overhead in highly mobile networks where location update is frequent, because each node has to send location updates to its network-wide distributed location servers. Zigzag-line message transmission also contributes to increased message overhead and search latency. If all nodes in an order-1 square move out, nobody will store forwarding pointer for them, causing search failure.

Many similar algorithms, for example, DLSP (Chen *et al.*, 2006), HLS (Kieb *et al.*, 2004), and MLS (Flury and Wattenhofer, 2006), to name a few, have been proposed on the basis of hierarchical division. In DLSP (Chen *et al.*, 2006), each actuator selects by a common hash function eight location servers in the eight neighboring squares in each level of a grid hierarchy. Location servers at different levels are updated at different rates. Actuator search starts at lowest level and across the hierarchy until satisfied. If search fails, a new round of search starts from the failure point.

In HLS (Kieb *et al.*, 2004), each actuator selects by a common hash function one minimum partition cell in every cell at each level in the partition hierarchy to construct a personalized tree. Location (pointer) update is along the downward path from the root to a leaf. Actuator search is done in the same tree from a leaf upward to the root. It may be satisfied early if the search path intersects the update path before the root. MLS (Flury and Wattenhofer, 2006) is very similar to HLS in location update. However, it does not use upward path traversal but expanding ring search for target actuator.

REFERENCES

Abraham I, Dolev D, Malkhi D. "LLS: a locality aware location service for mobile Ad hoc networks". Proceedings of the Joint Workshop on Foundations of Mobile Computing (DIALM-POMC); Philadelphia, Pennsylvania, USA; 2004. pp. 75–84.

Amouris KN, Papavassiliou S, Li M. "A position based multi-zone routing protocol for wide area mobile ad-hoc networks". Proceedings of the 49th IEEE Vehicular Technology Conference (VTC); Houston, Texas, USA; 1999. pp. 1365–1369.

Basagni S, Chlamtac I, Syrotiuk VR, Woodward BA. "A distance routing effect algorithm for mobility (DREAM)". Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom); Dallas, Texas, USA; 1998. pp. 76–84.

Blazevic L, Buttyan L, Capkun S, Giordano S, Hubaux J-P, Le Boudec J-Y. "Self-organization in mobile ad hoc networks: the approach of terminodes". IEEE Commun Mag 2001;39(6):166–175.

Bose P, Morin P, Stojmenovic I, Urrutia J. "Routing with guaranteed delivery in Ad hoc wireless networks". Proceedings of ACM DIALM (LNCS 4325); Seattle, WA, USA; 1999. pp. 48–55.

Camp T, Boleng J, Wilcox L. "Location information services in mobile ad hoc networks". Proceedings of the IEEE International Conference on Communications (ICC); New York City, USA; 2002. pp. 3318–3324.

Chen Z, Cho M, Shin KG. "Design of location service for a hybrid network of mobile actors and static sensors". Proceedings of the 27th IEEE Real-Time Systems Symposium (RTSS); Rio de Janeiro, Brazil; 2006. pp. 202–211.

Das SM, Pucha H, Hu YC. "On the scalability of rendezvous-based location services for geographic wireless ad hoc routing". Comput Netw 2007;51(13):3693–3714.

Flury R, Wattenhofer R. "MLS: an efficient location service for mobile ad hoc networks". Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc); Florence, Italy; 2006. pp. 226–237.

Friedman R, Korland G. "Timed grid routing (TIGR) bites off energy". Proceedings of the 6th ACM International Symposium on Mobile and Ad Hoc Networking and Computing (MobiHoc); Urbana-Champaign, IL, USA; 2005. pp. 438–448.

Karumanchi G, Muralidharan S, Prakash R. "Information dissemination in partitionable mobile ad hoc networks". Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems; Nürnberg, Germany; 1999. pp. 4–13.

Kasemann M, Fubler H, Hartenstein H, Mauve M. "A reactive location service for mobile Ad hoc networks". Technical Report TR-2002-014. Mannheim, Germany: University of Mannheim; 2002.

Kieb W, Fubler H, Widmer J, Mauve M. "Hierarchical location service for mobile ad-hoc networks". ACM SIGMOBILE Mobile Comput Commun Rev 2004;8(4):47–58.

Ko Y-B, Vaidya NH. "Location-aided routing (LAR) in mobile ad hoc networks". Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom); Dallas, Texas, USA; 1998. pp. 66–75.

Krishnamurthy G, Azizoglu A, Somani AK. "Optimal location management algorithms for mobile networks". Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM); Dallas, Texas, USA; 1998. pp. 223–232.

Liao W-H, Tseng Y-C, Sheu J-P. "GRID: a fully location-aware routing protocol for mobile ad hoc networks". Telecomm Syst 2001;18(1–3):61–84.

Li J, Jannotti J, De Couto DSJ, Karger DR, Morris R. "A scalable location service for geographic Ad hoc routing". Proceedings of the 6th Annual ACM International Conference on Mobile Computing and Networking (MobiCom); Boston, Massachusetts, USA; 2000. pp. 120–130.

Li X, Santoro N, Stojmenovic I. "Localized distance-sensitive service discovery in wireless sensor networks". Proceedings of the 1st ACM International Workshop on Foundations of Wireless Ad Hoc and Sensor Networking and Computing (FOWANC); Hong Kong, China; 2008. pp. 85–92.

Li X, Santoro N, Stojmenovic I. "Localized distance-sensitive service discovery in wireless sensor and actor networks". IEEE Trans Comput 2009;58(9):1275–1288.

Liu D, Jia X, Stojmenovic I "Quorum and connected dominating sets based location service in wireless ad hoc and sensor networks". Comput Commun 2007;30(18):3627–3643.

Liu D, Jia X, Stojmenovic I. "Distributed location service by using hierarchical rings for large scale Ad hoc networks"; 2008. In preparation.

Liu D, Stojmenovic I, Jia X. "A scalable quorum based location service in ad hoc and sensor networks". Proceedings of the 3rd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS); Vancouver, Canada; 2006. pp. 489–492.

Melamed R, Keidar I, Barel Y. "Octopus: a fault-tolerant and efficient ad-hoc routing protocol". ACM Wirel Netw 2007;14(6):777–793. Available at http://www.springerlink.com/content/jp0675w8745x15mt.

Ni S-Y, Tseng Y-C, Chen Y-S, Sheu J-P. “The broadcast storm problem in a mobile ad hoc network”. Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom); Seattle, Washington, USA; 1999. pp. 152–162.

Prakash R, Haas ZJ, Singhal M. “Load balanced location management for mobile systems using dynamic hashing and quorums”. Technical Report TR-UTDCS-05-97. Dallas: The University of Texas at Dallas; 1997.

Prakash R, Singhal M. “A dynamic approach to location management in mobile computing systems”. Proceedings of the 8th International Conference on Software Engineering and Knowledge Engineering (SEKE); Lake Tahoe, Nevada, USA; 1996. pp. 488–495.

Stojmenovic I. “A scalable quorum based location update scheme for routing in ad hoc wireless networks”. Technical Report TR-99-09. Ottawa, Canada: SITE, University of Ottawa; 1999.

Stojmenovic I. “Home agent based location update and destination search schemes in ad hoc wireless networks”. In: Zemliak A, Mastorakis NE, editors. Advances in information science and soft computing: WSEAS Press. Technical Report TR-99-10. Canada:, SITE, University of Ottawa, Greece; 2002a. pp. 6–11.

Stojmenovic I. “Location updates for efficient routing in ad hoc networks”. In: Stojmenovic I, editor. Handbook of wireless networks and mobile computing: Wiley, New York; 2002b. pp. 451–471.

Stojmenovic I, Lin X. “Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks”. IEEE Trans Parallel Distrib Syst 2001;12(10):1023–1032.

Stojmenovic I, Liu D, Jia X. “A scalable quorum based location service in ad hoc and sensor networks”. Int J Commun Netw Distrib Syst 2008;1(1):71–94.

Stojmenovic I, Ruhil AP, Lobiyal DK. “Voronoi diagram and convex hull based geocasting and routing in wireless networks”. Proceedings of the 8th IEEE Symposium on Computers and Communications (ISCC); Antalya, Turkey; 2003. pp. 51–56.

Stojmenovic I, Ruhil AP, Lobiyal DK. “Voronoi diagram and convex hull based geocasting and routing in wireless networks”. Wirel Commun Mobile Comput 2006;6(2):247–258.

Stojmenovic I, Russell M, Vukojevic B. “Depth first search and location based localized routing and QoS routing in wireless networks”. Proceedings of the IEEE International Conference on Parallel Processing; Toronto, Canada; 2000. pp. 173–180.

Stojmenovic I, Seddigh M, Zunic J. “Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks”. IEEE Trans Parallel Distrib Syst 2002;13(1):14–25.

Su W, Lee SJ, Gerla M. “Mobility prediction in wireless networks”. Proceedings of the IEEE Military Communications Conference (MILCOM); Los Angeles, California; 2000. pp. 491–495.

Takagi H, Kleinrock L. “Optimal transmission ranges for randomly distributed packet radio terminals”. IEEE Trans Commun 1984;32(3):246–257.

Tchakarov JB, Vaidya NH. “Efficient content location in wireless Ad hoc networks”. Proceedings of the 5th IEEE International Conference on Mobile Data Management (MDM); Berkeley, California, USA; 2004. pp. 74–85.

Viana AC, De Amorim MD, Fdida S, Viniotis Y, De Rezende JF. “Easily-managed and topology-independent location service for self-organizing networks”. Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc); Urbana-Champaign, IL, USA; 2005. pp. 193–204.

Wu K, Harms J. “Location trace aided routing in mobile Ad hoc networks”. Proceedings of the 9th International Conference on Computer Communications and Networks (ICCCN); Las Vegas, NV, USA; 2000. pp. 354–359.

Woo SCM, Singh S. “Scalable routing in ad hoc networks”. Wirel Netw 2001;7(5):513–529.

Xue Y, Li B, Nahrstedt K. “A scalable location management scheme in mobile Ad-hoc networks”. Proceedings of the 26th Annual IEEE Conference on Local Computer Networks (LCN); Tampa, Florida, USA; 2001. pp. 102–111.

Yang J, Li X, Nayak A, Stojmenovic I. "Integrated location service and geographic routing to a mobile sink in wireless sensor networks". 2009. A manuscript.

Yu F, Park S, Lee E, Kim S-H. "Quorum Base Sink Location Service for Geographic Routing in Arbitrary Irregular Wireless sensor networks"; 2009. Manuscript.

Zhang R, Zhao H, Labrador MA. "A scalable and energy efficient sink location service for large-scale wireless sensor networks". Ad Hoc & Sens Wirel Netw 2007;4(4):289–320.

Chapter 9

Coordination in Sensor, Actuator, and Robot Networks

Hai Liu[1], Veljko Malbasa[2], Ivan Mezei[2], Amiya Nayak[3], and Ivan Stojmenovic[2,3]

[1]*Hong Kong Baptist University, Hong Kong, P.R. China*
[2]*Faculty of Technical Sciences, University of Novi Sad, Serbia*
[3]*School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5*

Abstract

This chapter surveys the existing representative work in both sensor-actuator and actuator-actuator coordination. Sensor-actuator coordination deals with establishing data paths between sensors and actuators, and can be used for sensor deployment. Actuator-actuator coordination includes robot coordination for sensor placement, dynamic task allocation, selecting best robot to respond to reported event, robot dispersion, boundary coverage, and fault-tolerant response. In the coordinated actuator movement problem, actuators are moved to desired locations to save energy in long-term communication tasks where the traffic is sufficiently regular and large in volume to warrant nodes expending energy for moving. A recent study on coordination among flying robots is introduced in the end.

9.1 SENSOR-ACTUATOR COORDINATION

The collaborative operation of sensors enables the distributed sensing of a physical phenomenon. In wireless sensor networks (WSNs), the sink (base station) performs the functions of data collection, processing, and coordination. In wireless

Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication. By Amiya Nayak and Ivan Stojmenovic. Copyright © 2010 John Wiley & Sons, Inc.

sensor actuator networks (WSANs), both sensor-actuator and actuator-actuator coordination are required. After sensors detect an event that has occurred in the environment, the event data is processed (e.g., aggregated with reports from nearby sensors) and transmitted to the actuators, which gather, process, and eventually reconstruct the characteristics of the event. The process of establishing data paths between sensors and actuators is referred to as *sensor-actuator coordination* (Melodia *et al*., 2007). Sensor-actuator coordination provides the transmission of event features from sensors to actuators. Sensors and actuators coordinate also for some other tasks, such as sensor placement or improving connectivity.

Akyildiz and Kasimoglu (2004) did the first comprehensive analysis on both sensor-actuator and sensor-sensor coordination. There are few main requirements on the communications in sensor-actuator coordination (Akyildiz and Kasimoglu, 2004). The communications between sensors and actuators in WSANs require energy efficiency to prolong the lifetime of the network. In some real-time applications, for example, detection of fire, the communication traffic is typically delay sensitive. Therefore, sensor-actuator communication should support real-time traffic in these applications. The sensor-actuator communication is also required to ensure ordering of event data reported to the actuators. For example, suppose there are two sensors reporting two different events to a common actuator. These two events may need to be delivered in the sequence in which the events are detected so that the appropriate corresponding actions on the environment are taken. Synchronization among sensors is desirable when different sensors report an event to a common actuator, such that the actuator acts once in the entire event region. Sensors are required to track the event and use this information to determine the set of actuators to report the sensed phenomena.

If there are multiple actuators in the network, a natural question in sensor-actuator coordination is which actuators the sensed phenomena will be reported to? The problem is to select actuators to which the sensors will send their data. In an event-driven partition, only sensors located inside the event area are reporting. In Figure 9.1, the event area is a circle. Sensors are partitioned according to the actuator they report to, with actuators serving as roots of the corresponding data reporting trees. Data-delivery trees are created to provide the required reliability with minimal resource consumption (Melodia *et al*., 2007). These clusters may be formed in such a way that the event reporting time from the sensors to the clusterhead (CH) is minimized, the sensors report data to the CH via the minimum energy paths, or the action region of the actuators can cover the entire event area (Akyildiz and Kasimoglu, 2004).

Suppose B is the maximum allowed time delay between detecting an event by the sensors and receiving the event reports by actuators. A data packet is said to be "unreliable" if it does not meet the latency bound B when it is received by an actuator. The data packet is said to be "reliable" otherwise. Let r denote the ratio of reliable data packets over all the packets generated in a decision interval. The sensor-actuator coordination problem discussed in Melodia *et al.* (2007) is to establish data paths from each sensor to the actuator, such that the reliability

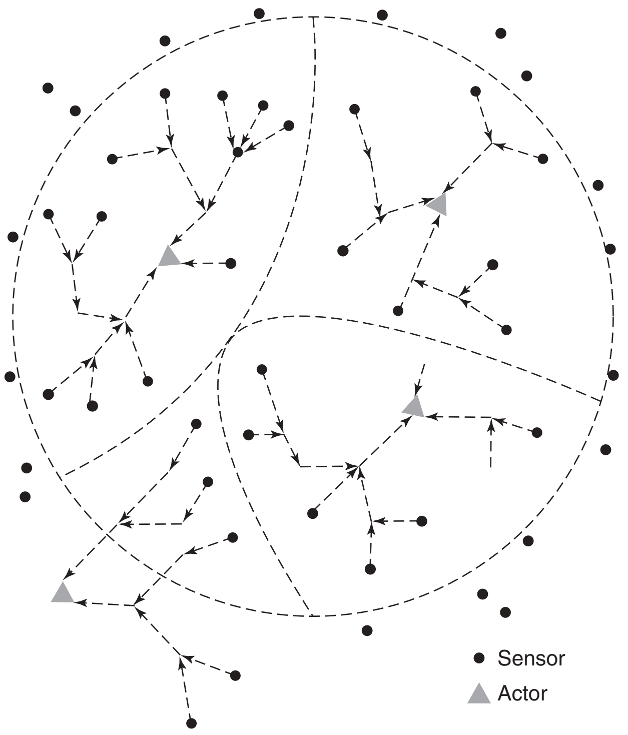


Figure 9.1 Reporting trees from sensors to multiple actuators.

ratio r is above the required threshold r_{th}, and the energy consumption associated with data-delivery paths is minimized.

Event-driven partitioning is a twofold problem: (i) select the optimal subset of actuators to which sensors will report data and (ii) construct the minimum energy data-delivery trees toward those selected actuators to meet the required threshold of reliability ratio. The union of all trees rooted at the actuators implicitly partitions the set of sensors in the event area (Fig. 9.1). The problem was formulated as an integer linear programming (ILP), which can be solved with moderate size (up to 100 nodes) (Melodia *et al.*, 2007).

A distributed algorithm was further proposed in Melodia *et al.* (2007) assuming that each sensor is aware of its position and the positions of its neighbors and a number of candidate actuators, and that the network is synchronized. Sensors are idle if there is no reporting (lack of events). When an event is detected, they enter the start-up phase of determining the actuator for reporting (and therefore sensor partitioning and creation of initial data reporting trees), and finding an initial path to one of actuators. Sensors on the created paths also move to start-up phase if they were in idle state. The initial task is equivalent to the *anycasting* problem, which was discussed in Chapter 5 (including the solution proposed in Melodia *et al.* (2007)). Sensors do not change the actuator they report to, but may change their initially created data reporting paths to achieve the desired level of reliability while minimizing the energy cost for that level. The basic idea for the modification of data reporting trees (Melodia *et al.*, 2007) is to increase the delay and reduce the energy consumption when the reliability ratio is high and reduce the delay at the expense of energy consumption when the reliability ratio is low. In each decision interval, each actuator computes the reliability ratio r of reliable data packets over all the packets and periodically broadcasts its value. Upon receiving feedback messages from the actuators, a sensor in the start-up state may enter speed-up or aggregation states, to either reduce delay (by sending the packets to the neighbor that is closest to the destination but within the transmission range) or increase delay but reduce energy consumption by sending the packets to the closest neighbor, which is in the same data-delivery tree. An improved algorithm for constructing delay bounded energy optimal data reporting trees, when data are aggregated on the way to actuator, is described in Chapter 7. It assumes that the delay for reporting is proportional to the number of hops on the path to the actuator. It is based on the estimate that the time for gathering reports from children nodes and aggregating them, and transmitting to the parent node is similar for each node. A different model is treated in Li *et al.* (2007), which assumes that the delay at each node is proportional to the number of children (plus one to also account for communicating with the parent node). This is justified by the delay in gathering the information from each child, which needs to be done sequentially, since collisions must be avoided.

A real-time communication framework for WSAN was proposed to support event detection, reporting, and actuator coordination in Ngai *et al.* (2006). Upon detecting an event, sensors report the event to other sensors nearby (sensor-to-sensor) in up to k hops. Replies with data aggregation (mean of known values) are received to verify the event. A report is then sent to the nearest actuator, which informs other actuators nearby, to select the closest one that should have the priority to respond. After the actuator that responds leaves for action, sensors that were closest to it will search for a new closest actuator by flooding requests until it reaches the sensors assigned to the neighboring actuators. The details of sensor-actuator and actuator-actuator communication are not given.

9.2 TASK ASSIGNMENT IN MULTIROBOT SYSTEMS

Multirobot systems (MRS) are well studied in literature and several survey papers, taxonomies, and book chapters exist on the subject (e.g., Dudek *et al.*, 2002; Farinelli *et al.*, 2004; Parker *et al.*, 2008). The focal point of the majority of MRS-related papers is on general communication, coordination, and cooperation. Networked robotics is an extension of MRS where communication between entities is fundamental to both cooperation and coordination, and hence the central role of the network (Bekey and Yuh, 2008; Kumar *et al.*, 2008). Applications of networked robots include coupling to perform locomotion and manipulation tasks, and coordinating to perform exploration, mapping, search and reconnaissance

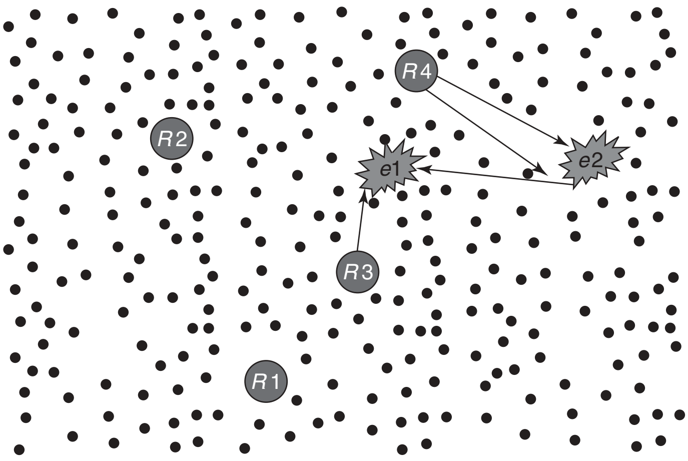


Figure 9.2 Monitored area with two concurrent events and two possible assignments.

tasks, or gaming (e.g., robot soccer). They can also perform independent tasks that need to be coordinated (e.g., hole drilling or welding).

We also use the term *wireless sensor and robot network* (WSRN). It addresses the special case when robots are used as mobile actors within sensor networks. An illustration is given in Figure 9.2, showing sensors, four robots, and two events. Wireless sensor and robot networks are characterized by real-time response constraints and sensor-sensor, sensor-robot, and robot-robot coordination. Sensor-robot coordination provides information about the event by means of transmitting data from sensors to robots. Mobile robots pose significantly harder challenges because sensors cannot easily maintain the position of nearest robot to report. In the absence of such information, sensors need to flood their event information, which drains energy by a lot of wasted communication. To reduce that, robots need to proactively maintain their location information by sending periodic messages to certain areas or directions (see Chapter 8), so that sensors may report efficiently when event occurs. This is a very difficult task. In Selvaradjou *et al.* (2006), the problem is resolved by assuming that area is partitioned into grids, with at least one actor in each (thus large number of them is available and is nearby any possible event), and sensor grid leaders ("agents") being further organized into clusters.

A WSAN operates in the loop of event sensing, data communications and actuator actions. Mobile self-controlled actuators are often referred to as *robots*. In WSANs, once the event has been detected and reported from sensors to the actuators (robots), the actuators coordinate to reconstruct it, to estimate its characteristics, and to make a collaborative decision on the action to follow and how to perform it. This overall process is an example of *actuator-actuator coordination* (Akyildiz and Kasimoglu, 2004). After reaching an agreement on the actions needed, a fundamental problem in the actuator-actuator coordination is to decide which actuators should execute which actions, such that the overall performance of the actuators is maximized. The problem is referred to as the *Task Assignment* problem (Melodia *et al.*, 2007) in WSANs.

Each event in WSANs is assumed to be reported to one of robots. In case of multiple (possibly concurrent) events, several tasks are to be distributed among robot team members. For example, in fire monitoring scenario, sensors detecting fire route information to robots that need to decide which of them should act to exhaust the fire. Sometimes tasks need more than one robot to be successfully accomplished (e.g., larger fire). In the case of *multiple* events (e.g., fire on multiple locations, or multiple sensors deployment), concurrency and time constraints could decide the action. If there are no time limits then all multiple event problems can be treated as iterated version of single event problem. The difference then is whether or not these events are known in advance, or unpredictable. In the former case, a single robot could be assigned to handle multiple events in the sequence. For example, in Figure 9.2 robot $R4$ could respond to event $e2$ followed by event $e1$. Alternatively, it could only handle event $e2$ while robot $R3$ responds to event $e1$. In summary, there are single/multiple task requiring single/multiple robots with/without strict time constraints for the execution.

These general problem statements allow for considering many specific instances. There are few generic approaches to solve the corresponding problems. Depending on decision making process, the task assignment solutions can be classified as *centralized*, *localized*, and *market-based* solutions. In centralized solutions, decisions are made at a single robot or the sink (base station) after gathering all the necessary information. In localized solutions, decisions of each robot are based on the information available in its direct neighborhood and incoming request. Compared with centralized solutions, localized solutions have low computation and communication cost, good scalability, fault tolerance, and are fast. However, it is hard to find optimal decisions in localized solutions due to limited information. In distributed market-based solutions, robots communicate with certain number of other robots in the smaller or larger vicinity in order to negotiate to resolve lack or excess of actions in locally made decisions. The objective is to achieve better response efficiency of individual robots and robot network as a whole. In market-based solution, robots are working as free agents to maximize their individual profit. They can negotiate and bid for tasks by means of *auctions*, which is the main tool to make decisions. The communication involved in the auction process may exceed strict neighborhood information, and be compromised toward a semilocalized behavior.

9.3 SELECTING BEST ROBOT(S) WHEN COMMUNICATION COST IS NEGLIGIBLE

Most existing solutions referring to multirobot coordination for single or multiple events, single or multiple robot, single or multiple task to each robot and so on, are centralized. One of robots, or a central entity, gathers all the information from other robots and makes a decision. Communication cost for gathering information in case of multihop robot networks is rarely considered. The decision maker can theoretically make an optimal decision by gathering necessary information with little cost because communication cost is assumed negligible. Indirectly (since no details of communication protocols used are given), a *complete graph* (where each robot is within communication distance to any other robot) is assumed. Centralized solutions usually define coordination problem as an ILP problem. Main advantage of a centralized solution is that, theoretically, optimal solution can be found. However, centralized solution features high computation and communication overhead, bad scalability and slow responsiveness. Moreover, the actual cost for communicating is ignored, especially for large robot networks. It is further not clear how robots communicate if the graph is not complete one. Centralized solution also has low fault tolerance if leader is malfunctioning in any way.

For static actors, Melodia *et al.* (2007) stated a mixed integer nonlinear program (MINLP) formulation to maximize average remaining energy of (multi) robots selected to perform a task, under the constraint of meeting action completion bound. Shah and Meng (2007) assumed the presence of a global unit, which receives updates on the work status of each robot for each task (beginning, completing, in-trouble or time-out). When a robot needs help for the task allocated, global unit is able to select one or more for assistance. The selection is based on an auction-based method that considers the capability, distance from current task location, robot availability, and number of tasks in local queue of robot. The article by Selvaradjou *et al.* (2006) showed that finding the optimal assignment of (single) robots toward the events with the objective of minimizing the overall movement of robots, with deadline and energy constraints, is NP-complete. They formulate MINLP. Such a formulation assumes *a priori* knowledge of events and tasks, and finds a tour for visiting events for each robot. They also propose inter and intrazone-based localized heuristics for finding the near optimal schedule of the robots to the appropriate events. In intrazone scheduling, agent sensor nodes compute optimal schedule for robots within their zones. In interzone scheduling, CHs (of clusters of agents) find optimal schedules for all robots within their clusters. The proposed heuristic is to assign the task with the earliest deadline to the nearest robot that can perform task within the specified deadline, and having enough energy.

There are no multiple robots assigned to a single event, and no multiple assignments are given to a single robot. The problem is recognized as an instance of *optimal assignment problem* and is covered in detail in Gerkey and Mataric (2004). It discusses centralized solutions where task and robot information are all gathered at a single place, and the number of tasks does not exceed the number of robots. If the number of robots equals the number of tasks, this scenario is an instance of *classical assignment problem* (also known as *linear sum assignment problem*). If robot energy is limited, the goal is to find such an assignment that minimizes overall energy spent on mobility (for all robots moving to their corresponding events), assuming that mobility energy is proportional to distance traveled. The energy needed to perform tasks is not in the optimization function, but could be added. The problem can be expanded to an instance of the *assignment problem with side constraints*, if total spent energy must be within the energy budget for every robot. For tasks with real-time constraints, time-bound requirements (time needed to arrive to an area plus the time to perform the task) could be added. If the number of tasks exceeds the number of robots, the algorithm can be repeated in rounds by applying them on tasks sorted by their deadlines and assigning one to each robot in each round. Alternative approaches may include assigning several tasks to a single robot at once, with sequential access to them and tour-like visits from event to event. Finally, a greedy algorithm may even assign tasks in sequence, choosing one robot (among available ones) for one task.

The problem formulations considered so far in literature did not include network degradation issues due to possible energy depletion of some robots. The best robot for a given task may minimize energy needs to the robot team, but could also lead to its own inability to perform any other task, which may also impact the ability of the team to provide support to sensors. Therefore, some load balancing mechanisms may be needed to address the issue. The best robot can be determined by negotiation among robots, with modified goals, to maximize the time the first robot looses all its energy. Alternate definitions are possible, such as the time the robot network becomes partitioned, or the portion of monitored area or reaction time by the team becomes unacceptable. One possible load balancing strategy is to modify the cost function for any robot, to include the remaining energy. Upon receiving an action request to serve an event, each robot B calculates d/E, where d is distance of robot B from the event, and E its current remaining energy (alternatively, energy after performing the task may be considered). One that meets real-time constraints and minimizes this ratio will take over the task.

9.4 SELECTING BEST ROBOT(S) WITH NONNEGLIGIBLE COMMUNICATION COSTS

If the robot network is disconnected then sensor nodes may be used to connect some robots. Sensors and robots may both participate in the same flooding protocol, with different transmission ranges being available to them. Robots may apply smaller backoff waiting times for retransmissions, so that they get a priority over sensors for retransmitting. If the transmission region (the set of its neighbors) of a particular sensor or robot is not covered by all received transmissions before the time runs out, the node retransmits the message. For simplicity, we assume here that the robot network is connected, and discuss further only robot-robot communication.

Solutions for robot-robot coordination described in this section do not depend on the particular environment served by networked robots. One such environment of interest to us are wireless sensor and robot networks (WSRNs), as an extension of MRS. Wireless sensor and robot networks consist of sensors and robots linked by wireless medium to perform distributed sensing of the physical world, processing of sensed data, making decisions, and acting upon sensed events (Fig. 9.3).

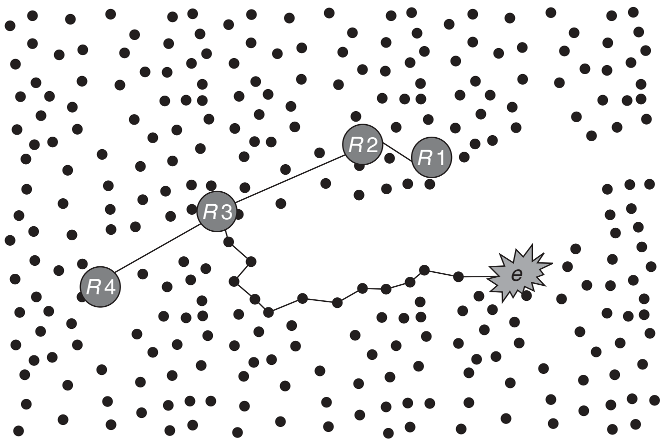


Figure 9.3 Reporting events from a sensor to nonnearest actuators due to void area.

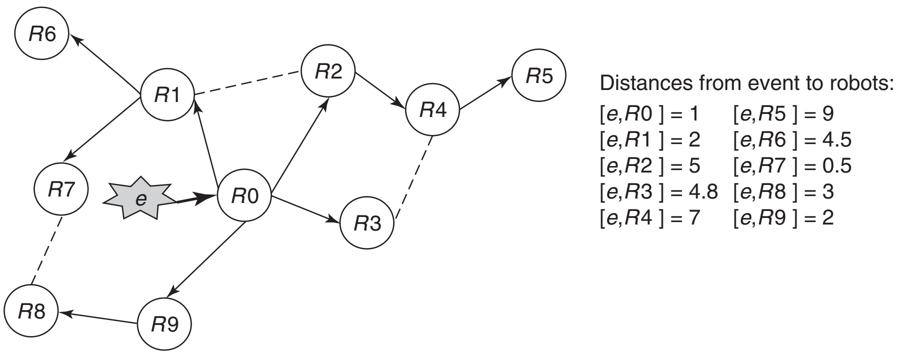
We will illustrate our problem statement using this scenario. In many cases, the robot that receives the report may itself be the best candidate for responding. However, a remote robot could receive the report. Upon event occurrence (e.g., a fire, or the failure of a sensor), sensors detect events and route information to *any* robot in the vicinity. However, it may not be the closest one.

In Figure 9.3, $R1$ consults $R4$ and $R2$, and learns from $R4$ about the closest robot $R3$ to the event. In the example in Figure 9.3, $R3$ received report because there is a sensor void area between the event and closest robot $R1$. $R1$ is able to act but sensors were not able to report the event directly to it. $R3$ initiates the bidding process and discovers the nearest robot $R1$, which is then assigned to the task.

Localized and distributed solutions utilize spreading all decision making and planning responsibility among robots. We consider here, the *multihop* unit disk graph (UDG) scenarios, where the communication graph is not complete. Robots use only locally available information to make their decision. Good scalability and fault tolerance are the main advantages. Proposed solutions are normally close to the optimal one. However, decisions made based on the local information can be sometimes highly suboptimal.

The article by Melodia *et al.* (2007) described a localized solution for actor-actor coordination (for single robot single task assignment scenario), based on simple *auction protocol* (SAP). The auctioneer robot floods event information to all other robots, and each actor (robot) reports back to the originating actor (by a separate routing task) the offer to provide service and the cost of doing it. Bids are sent from robots that are able to serve the given area. If blind flooding is used, each robot retransmits the request upon receiving it for the first time, and ignores it afterwards. In an intelligent flooding (see Chapter 2), only nodes from a backbone are transmitting, and only if they have neighbors in need of a message. For large robot networks, this protocol incurs unacceptable delay in selecting the best robot, while the best responding robot is expected to be near the event.

Robot selection can be limited to certain local neighborhoods, even for SAP (Mezei *et al.*, submitted for publication). A localized solution, based on market paradigm, called *auction aggregation protocol* is further proposed (Mezei *et al.*, submitted for publication). The bidding process spreads to neighboring robots until no improvement can be envisioned within the k-hop neighborhood of a robot, which analyzes if any other remote robot could provide better service than the best service it is aware of. If not, it stops search process and responds back to its "parent" robot with best possible recommendation it has. Instead of using separate routing tasks, the constructed tree can be used for reporting back. The protocol has tree "expansion" and tree "contraction" phases. Tree expansion starts from collecting robot $R0$ by creating a tree rooted at $R0$ (Fig. 9.4). Retransmissions create a response tree. Each node, with retransmission, includes the ID of its parent robot in the message, so that robots can locally decide whether or not they are leaves in the created tree. Note that each node selects only one parent, in case of multiple received bids (e.g., $R8$ joins only $R9$ and $R4$ joins only $R2$). They become leaves if they do not retransmit the bid or do not hear any other robot listing them as their parent. Leaf nodes start responding back to robots, with the best cost they are aware of. This is in fact auction aggregation and thus reduces the number of messages in the bidding phase. Each intermediate node waits to hear from all neighbors, which declared it as a parent thus becoming a local collector. After hearing, they select the best cost and report further toward the collector. The collector at the end decides which robot is the best to perform the required action, and routes the decision to that robot. In the example in Figure 9.4, robots $R3$, $R5$, $R6$, $R7$, and $R8$ are leaves in the created tree, and return their bids to their parent nodes. $R9$ returns to its parent $R0$, its own bid 2 as the best it is aware of. Similarly, $R4$ returns its own bid 7 to $R2$, which returns its own bid 5 to $R0$. $R1$ returns $R7$ as the best bidder to its parent $R0$. The root node ($R0$) then selects the best bid (in this case from $R7$) from four received offers,



Distances from event to robots:

[e,$R0$] = 1	[e,$R5$] = 9
[e,$R1$] = 2	[e,$R6$] = 4.5
[e,$R2$] = 5	[e,$R7$] = 0.5
[e,$R3$] = 4.8	[e,$R8$] = 3
[e,$R4$] = 7	[e,$R9$] = 2

Figure 9.4 Auction aggregation protocol for selecting the best robot.

from $R1$, $R2$, $R3$, and $R9$ respectively, and delivers the task to $R7$ along the created path $R0$-$R1$-$R7$. This version of auction aggregation protocol will be designated as simple auction aggregation protocol (*SAAP*). In case of limited flooding (only up to k-hop neighbors from the collector), it is called *k-SAAP* (Mezei *et al.*, submitted for publication). The difference between *k-SAAP* and *k-SAP* (and similarly between *SAP* and *SAAP*) is that individual bids are aggregated at intermediate nodes, instead of routing all of them back to the collector robot.

The algorithm can be further refined by providing autonomy in retransmitting decisions to individual robots. In *k-SAAP*, the receiving robot will retransmit only if it is at distance of $<k$ hops from the bidding robot. The *k-hop auction aggregation protocol* (*k-AAP*) protocol (Mezei *et al.*, submitted for publication) applies k-hop neighborhood around current robot. Each robot is assumed to already know the position, cost, and availability of all its neighboring robots up to k hops away. One simple way is periodic diffusion of its local information [including the status of its (k – 1)-hop neighbors] to its neighbors ("hello" message), or piggybacking it to data messages. In *k-AAP* (Mezei *et al.*, submitted for publication), the robot that received the bid (and the best learned cost C associated with previous senders on the path from the bidding robot to it) will compare C and the cost of providing service by its k-hop neighbors. It will retransmit the message only if at least one of its own k-hop neighbors has cost $<C$. Otherwise it will not retransmit, and will perform auction aggregation by returning a response message to its parent node on the search path. In the simplest version, 0-*AAP* protocol (for $k = 0$), the receiving robot declares itself as the selected one, corresponding to the use of 0-hop knowledge (no knowledge at all). In the 1-*AAP* protocol, collecting robot $R0$ will retransmit if any of its neighbors has a lower cost. In the example in Figure 9.4, this still results in no transmission from $R0$. In the k-*AAP*, $R0$ will retransmit if any of its k-hop neighbors has a lower cost. In Figure 9.4, $R7$ is 2-hop neighbor of $R0$ with a lower cost, and $R0$ then retransmits in the 2-*AAP* version. After receiving the bid from $R0$, $R2$, $R3$, and $R9$ will not retransmit, while $R1$ will because $R7$ is its 1-hop neighbor. Every retransmitting node will include with the message the lowest cost it is aware of.

In the special case when cost metrics equals distance (from given robot to the event), another specialized algorithm may be used. We formalize it as Routing Toward the event with Face traversal encircling the event (*RFTTF*) algorithm (routing toward the event with face traversal encircling the event). It is in fact algorithmically equivalent to the *face traversal* algorithm described in Chapter 4. In the example in Figure 9.5, e is the event location. The *RTF* algorithm starts after the collecting robot V receives the task. *RTF* routes from V, using robot networks, toward event location e. Routing will end by traversing a face containing e. In Figure 9.5, face is $V \text{–>} A \text{–>} B \text{–>} C \text{–>} D \text{–>} F \text{–>} G \text{–>} A$. Face traversal makes a loop, which can be locally detected. The first node in the loop (A in Fig. 9.5) decides the best (closest) robot (B in this example). Face traversal requires planar graphs, for example, the Gabriel graph of UDG, which is introduced in Chapter 4.

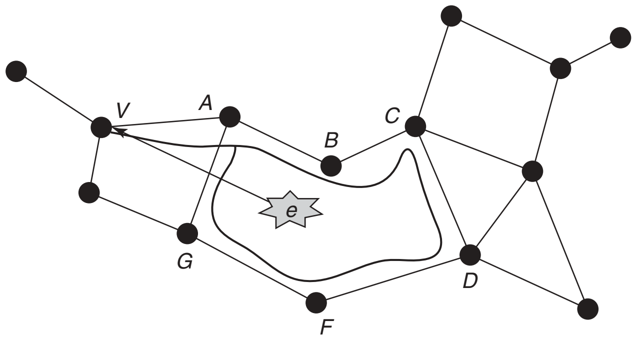


Figure 9.5 Face traversal algorithm over Gabriel graph of UDG.

9.5 DYNAMIC TASK ASSIGNMENT

In some applications, the dynamic assignment of tasks among robots is required. For instance, in a search-and-rescue mission of 40 robots, a preferred task assignment might be 30 robots to explore the environment, 2 to mark resources, and 8 to maintain a communications network (Ogren *et al.*, 2002). In general, assume that p_i robots need to be assigned to ith task, and that each robot can be assigned to only one task.

Four distributed algorithms were proposed by McLurkin and Yamins (2005) to assign swarms of homogenous robots to subgroups, each of which performs a different task to meet a specified global task distribution. In the simple *Random-Choice* algorithms, each robot chooses a given task with a probability proportional to the relative size of that task subgroup in the target distribution. In the above example, each node chooses tasks with probabilities 3/4, 1/20, 1/5, respectively. The *Random-Choice* algorithm does not require communication among robots and completes immediately. However, there is a high probability that it will fail to achieve acceptable target distribution if the size of swarms is small.

The next, the *Extreme-Comm* algorithm is another extreme for the communication overhead involved. In short, robots flood necessary information, including their IDs and task demands, until all robots learn all the information needed to make the same decision as a centralized algorithm. The tasks can be assigned in sorted order of robot IDs. More precisely, each robot broadcasts a message containing its ID and relays messages received from its neighbors. These messages propagate from each robot throughout the network. Thus, each robot is able to build a complete list of all other robots in the swarm. Suppose the given target distribution vector is $\{p_1, p_2, \ldots, p_m\}$ and the total number of robots is n. Each robot determines its relative position X in the ID list and selects a task p_i, such that $p_1 + \cdots + p_{i-1} \leq X < p_1 + \cdots + p_i$. The algorithm runs fast and outputs accurate solutions since IDs of all robots are available to each robot. However, the algorithm requires a large amount of interrobot communications.

The third, the *Card-Dealer* algorithm breaks task assignment into a series of stages. Wave propagation is used to learn task leader (select the robot for the next task) in each round. In a given round, the selected robot is the one with the smallest ID that did not determine its task yet. To learn that robot, each robot (only robots that are not yet assigned tasks "compete") broadcasts its ID to its neighbors repeatedly until it receives a similar broadcast message that originated from a robot with a smaller ID. It switches to retransmitting this smaller ID instead. Robots that do not compete for the next task will only retransmit the received message with the smallest ID. The procedure continues until the smallest ID in the network is the only remaining one, and all robots are aware of it.

Finally, in the *Tree-Recolor* algorithm, wave propagation is applied to learn one leader only (in one round), creating a spanning tree rooted at the leader, who decides the roles of each robot and communicates it to them. More precisely, in one round of wave propagation, the node with the smallest ID becomes the leader. It initiates a flooding (where each receiving robot will retransmit the message exactly ones) to construct a spanning tree of the network with itself as the root. Flooding is covered in Chapter 2. Each robot establishes a pointer to the parent robot, the one from which the first copy of flooded message is received. In this way, all robots are joined in a single tree with the leader as the root. The root (leader) learns all the tasks, makes decisions, and communicates roles (tasks) for each robot using communication over the constructed tree. For proper allocation, the internal nodes of the tree should store the number of their descendent nodes (robots). This can be found by reverse broadcasting that starts from leaves and continues toward the root. Leaves (level 1 robots) send messages to level 2 robots, which learn the number of their children, and forward the information to their parents. Each internal node calculates the sum of children node counts to find its own count of number of nodes in its own subtree. Using these counters, and the list of tasks to be allocated, the root can start distribution by delivering the appropriate number of tasks to each of its subtrees. This process continues recursively from each child node, starting with the corresponding allocated list of tasks. The allocation messages spread along the spanning tree from the source to all robots in the network. Communications overhead is lower than in the previous two solutions.

9.6 DEPLOYING SENSORS TO IMPROVE CONNECTIVITY

The algorithm in Seah *et al*. (2006) builds a system of networked wireless sensors, and actuators carried on mobile robots that are able to operate in all kinds of terrain. Robots (actuators) move to fill the communication gaps to enhance connectivity while some nodes serve as landmark nodes to help robots search the targets. The algorithm is motivated by a scenario of a large indoor environment with rooms. Sensors are scattered in the area but doors and windows prevent connectivity and sufficient coverage. After applying the localization algorithm (Wong *et al*., 2005), every sensor node keeps an n-tuple of its hop count distances from all the n landmarks. Each sensor periodically broadcasts beacon messages, which carry this n-tuple, to its neighbors, and to nearby actuators. The newly deployed sensors will broadcast a localization packet to update the n-tuple hop counts of sensors around the region. If a mobile actuator receives largely different hop counts (toward landmarks) from sensors around it, it identifies the area as a critical one, and tries to find a suitable spot to bridge the gap by deploying a new sensor.

A swarm-like model for movement direction of robots was proposed (SEAH *et al.*, 2006) to cooperatively search for communication gaps and targets in the environment. Each robot (actuator) periodically broadcasts its ID and heading to its neighbors. Each robot will adjust its heading only if its neighbors' IDs are higher than itself. The new heading is the heading of the neighbor with the closest higher ID plus 90° in the clockwise direction. The process is illustrated in Figure 9.6. The initial movement directions of four neighboring nodes are shown in Figure 9.6a. Suppose nodes make decisions in the order of 4, 3, 2, and 1. Since node 4 has the highest ID among its neighbors, it does not change its heading. The new heading of node 3 is the heading of node 4 plus 90° in the clockwise direction (Fig. 9.6b). Similarly, nodes 2 and 1 adjust their heading based on heading of nodes 3 and 2, respectively. The final orientations of nodes are shown in Figure 9.6d.

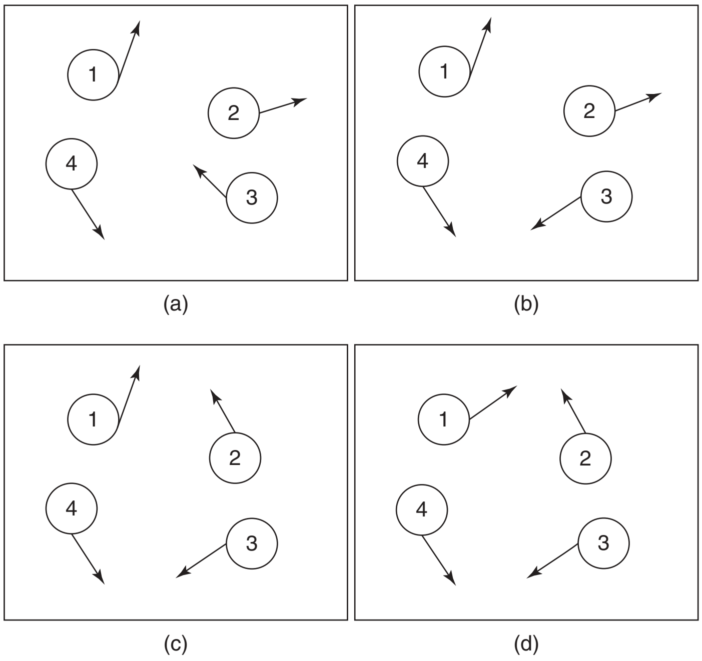


Figure 9.6 Selecting movement directions by actuators. (a) Initial movement directions. (b) Node 3 changes its heading. (c) Node 2 changes its heading. (d) Final movement directions.

In this problem statement, robots coordinate for sensor placement in the field. Chapter 10 describes a number of other techniques for sensor placement by robots, with or without robot coordination.

9.7 FAULT-TOLERANT SEMIPASSIVE COORDINATION AMONG ACTUATORS

A multiactuator/multisensor model was studied in Ozaki *et al.* (2007) to provide fault tolerance for WSANs. Each sensor node sends a sensed value to multiple actuators and each actuator receives sensed values from multiple sensor nodes. Such reporting model provides fault tolerance if any sensor or any actuator are faulty. A semipassive coordination protocol was proposed in Ozaki *et al.* (2007).

One actuator plays a role of a coordinator, called *primary* actuator, while other actuators are *backup* actuators in an event area. The protocol (Ozaki *et al.*, 2007) consists of three phases: broadcast phase, decision phase, and update phase. In the broadcast phase, if an event occurs, a sensor node sends a measured value to multiple actuators in the area. Due to collision and noise in wireless channels, some actuators may not receive the value correctly. Each receiving backup actuator sends a message with the sensed value and the ID of the corresponding sensor node to the primary actuator. Once the primary actuator receives a certain number of messages (greater than a predetermined threshold), it makes a decision on the measured value from the value set. It can be a majority value from a discrete set, or, in the case of continuous values, a function filter could be used to remove extreme values and compute the average of the remaining ones. The estimated value and other updates are then broadcasted to all the backup actuators. Upon receiving this message, each backup actuator stores the received update values and acknowledges the receipt to the primary actuator. The primary actuator (or one of primary actuators in case there are several of them) then makes a decision on possible action, including a confirmed update value. In this way, a backup actuator can obtain the value from the decision message even if it misses the previous message containing the original updates. Upon receiving the decision and updates, each backup actuator updates the state by using the value stored in the local storage. Finally, the primary actuator acts on the decision made by sending a message to an actuation device, which acts based on received instruction.

If the primary actuator is faulty, for example, the update messages are delayed, the backup actuators suspect the primary actuator to be faulty due to time-out and send a negative acknowledgement message to the primary actuator. Then, one of the backup actuators, for example, the backup actuator with the smallest ID, will take over the role of the primary actuator. The new primary actuator computes an interval value from the received sensed values and sends an update message to the backup actuators, exactly as the faulty primary actuator was supposed to do. The system will be recovered. No specific protocols were described in Ozaki *et al.* (2007) for flooding tasks involved in the algorithm.

9.8 DISPERSION OF AUTONOMOUS MOBILE ROBOTS

McLurkin and Smith (2004) studied dispersion of autonomous mobile robots. The problem is to disperse a large swarm of robots into a space of interest to increase their area coverage while maintaining their network connectivity. The main ideas of their algorithms are as follows. Robots move opposite to vector sum of their forces toward neighbors. Frontier robots move forward. To prevent disconnections and oscillations, and preserve connectivity with two children, leaf robots also preserve coverage of initial area and keep the near robots stationary while the frontier moves.

Two dispersion algorithms, called disperseUniformly and frontierGuidedDispersion, were proposed in McLurkin and Smith (2004). These algorithms are run alternately on the swarm of robots. The disperseUniformly is to spread robots evenly, while the frontierGuidedDispersion algorithm is to direct robots toward unexplored areas. In the disperseUniformly algorithm, the basic idea is to direct robots using boundary conditions to limit the dispersion. Physical walls and a maximum dispersion distance between any two robots, r_{safe}, are used as boundary conditions to help prevent the swarm from spreading too thin and fracturing into multiple disconnected components. Each robot is directed away from the vector sum of the relative positions $\{p_1, \ldots, p_c\}$ of its c closest neighbors. The magnitude of the velocity vector is:

$$v = -\frac{v_{\max}}{c \times r_{\text{safe}}} \sum_{i=1}^{c} p_i I_i,$$

where I_i is an indicator function $I_i = \begin{cases} 1 & \text{if } |p_i| \leq r_{\text{safe}} \\ 0 & \text{otherwise} \end{cases}$ and $v_{\max}$ is the maximum allowable velocity. The vector acts as virtual force and directs the robot away from its c closest neighbors. The experiment results have shown that using two closest neighbors ($c = 2$), works best in practice.

The frontierGuidedDispersion algorithm uses robots on the frontiers of the explored space to guide the swarm into unoccupied areas. The system goal can be achieved if all the frontier robots move along their optimal path and "pull" the rest of the swarm into their final positions. Each robot is supposed to have three statuses: *wall*, *frontier*, and *interior*. Wall robots are those robots that detect an obstacle. They create virtual neighbors at a symmetric position with respect to the wall, which serves as a virtual force against continuing forward movement. Frontier robots are those that have no neighbors and no walls in an angular range over 180°. That is, frontier robots are on the edge of an open space. Interior robots are those that are neither wall robots nor frontier robots. In the example in Figure 9.7, W denotes the wall robots, F the frontier robots, I the interior robots. The five circles denote the virtual neighbors created by the robots, which detect nearby walls.

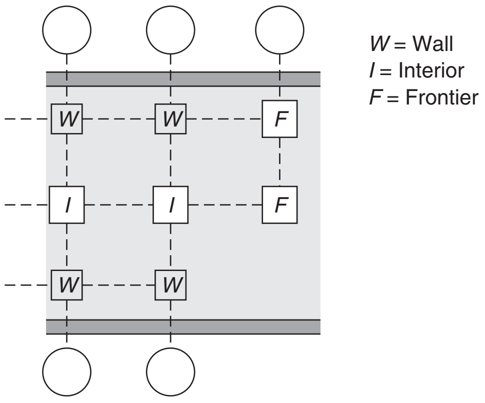


Figure 9.7 Wall, frontier, and interior robots.

Frontier robots guide the swarm into unexplored areas by propagating a gradient that forms a tree rooted at them. All robots then move away from their children in this tree. Leaves of the tree do not move to provide a route to the chargers or to mark previously explored areas. "Frontier trees" guide the swarm toward the frontier robots. Frontier robots "pull" the rest of the swarm behind them. However, any algorithm that is based on pulling robots over multiple hops can cause newly discovered frontiers to pull robots away from previously explored areas. This causes a frontier to reappear at the old location and pull the swarm back, creating oscillations, or fracturing the swarm and disconnecting robots from the initial positions (which could be places needed to charge robots). Instead, robots move away only from children in the frontier tree. To build a reliable (biconnected) network for fault-tolerant applications, robots should only move if they are in contact with at least two children in the frontier tree. This increases the min-cut of the network to two while the robots are dispersing, and helps deal with voids created by corners or robots heading home for charging.

9.9 DISTRIBUTED BOUNDARY COVERAGE BY ROBOTS

The distributed boundary coverage was studied by Correll *et al.* (2006) and Correll and Martinoli (2007a). They described self-organized embedded sensor/actuator networks for "smart" turbines. Robots are utilized to cover every point on the boundaries of all objects in an environment. The work is motivated by the inspection of the blades in the compressor section of a jet turbine. The process is currently performed by using borescopes, which is time consuming and costly. The narrow structure of the turbine motivates the use of extremely miniaturized robots. A swarm of miniature robots performs boundary coverage of blades in a jet turbine mock-up. The robots exploit the regularity of the environment and construct a spanning tree with the blades as vertices (Fig. 9.8). Edge traversal is achieved by a combination of dead-reckoning and navigation along way-points on a blade's boundary. Way-points can be determined by on-board

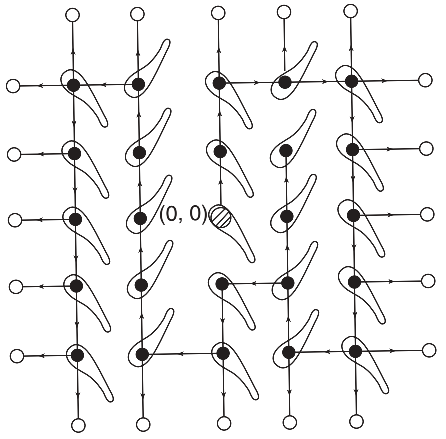


Figure 9.8 A 5 × 5 blade environment.

sensors, which are able to detect a blade's tips and to measure the curvature of the blade.

The spanning tree is explored by using depth first search (DFS) from the specified root. Edges are randomly selected from the unexplored candidates. An edge is said to be explored if two end points have been visited. If all edges of a node are explored, DFS returns to the parent of this node (backtracking). The algorithm terminates when the exploration packet returns back to the root. Note that DFS explores all possible edges, including nonnavigable ones ending at a wall. Figure 9.8 illustrates a possible spanning tree explored by the DFS, with a single robot traversing the blades. The central point (0, 0) is the root and white circles are nonnavigable vertices. Each edge is visited twice, once for exploration and once for backtracking. Since the sensor and actuator noise exists, a robot may miss a blade and continue exploration until the algorithm finds an unexpected reading, for example, a wall instead of a blade. Each robot is therefore assigned a portion of blades to apply DFS of the spanning tree of these blades. However, the division of search boundary among robots is not explained in Correll *et al.* (2006); Correll and Martinoli, (2007a).

9.10 CLUSTERING ROBOT SWARMS

A self-organized robot aggregation model was studied in Correll and Martinoli (2007b). The study is inspired by swarms of German cockroaches. Cockroaches move randomly through the arena and eventually stop and aggregate into clusters of different sizes. In each cluster, a cockroach can sense the presence of at least one other cockroach. Those clusters are not persistent and cockroaches might resume moving and leave the cluster. The average time for a cockroach to rest within a cluster increases while the size of the cluster increases (Jeanson *et al.*, 2004).

Let $p^{\text{leave}}(j)$ and $p^{\text{join}}(j)$ denote the probability a cockroach/robot leaves and joins a cluster of size j, respectively. These values are observed in Jeanson *et al.*

(2004). Let p_c denote the probability that a robot encounters another one at every time step of length T. p_c is estimated as follows: $p_c = (1/A_{\text{total}})v_r w_d T$, where A_{total} is the area of the arena, v_r the average speed of an individual, w_d the individual's detection width, that is, communication range of the robot. Therefore, the behavior of robots can be modeled as the Markov process. The average number of robots $N_j(k+1)$ in an aggregation of size j at time $k+1$, is as follows:

$$N_j(k+1) = N_j(k) + p_c N_{j-1}(k+1) N_s(k) p^{\text{join}}(j-1)j + p^{\text{leave}}(j+1)$$
$$N_{j+1}(k)j - p_c N_j(k) N_s(k) p^{\text{join}}(j)j - p^{\text{leave}}(j) N_j(k) j.$$

The first term $N_j(k)$ is the average number of robots in an aggregation of size j at time k. The second term indicates that a searching robot encounters one of the robots in a cluster of size $j-1$, and joins this cluster to form a cluster of size j. The number of searching robots is denoted by $N_s(k)$. The probability that a searching robot encounters one of robots in a cluster of size $j-1$ is $p_c N_{j-1}(k)$. The probability that this searching robot joins this cluster is p^{join} $(j-1)$. Similarly, the third term represents a robot leaving clusters of size $j+1$. The fourth term represents a searching robot joining clusters of size j, and the last term represents a robot leaving cluster of size j.

9.11 ROBOT TEAMS FOR EXPLORATION AND MAPPING

Howard *et al.* (2006) studied and experimented with a heterogeneous mobile robot team for exploration, mapping, deployment, and detection. The robot team, consisting of approximately 80 robots, has two classes of nodes: the mapper/leader robots and sensor mini-robots. All robots are equipped with 802.11b WiFi. The mission consists of two phases: exploration and mapping, and deployment and detection. In the exploration and mapping phase, the mapping subteam explores the environment and generates an occupancy grid map. In the deployment and detection phase, the occupancy grid map is used to compute a set of deployment locations, and simple sensor robots are deployed to these locations following guidelines from more capable robots.

Each mapping robot is equipped with a scanning laser range-finder, a color camera, and a unique coded fiducial, such that they can determine the identity, range, and bearing of nearby robots (Howard *et al.*, 2006). A pair of robots can determine their full relative pose, including range, bearing, and orientation, by exchanging such observations. Each mutual observation is transmitted to the remote operator console where the observation is added to the set of global constraints. A decentralized frontier-based approach (Yamauchi, 1997) was employed for exploration in Howard *et al.* (2006). The algorithm operates as follows. First, construct a local occupancy grid map using laser range data and local pose estimates. Then extract a list of discrete frontiers, that is, the boundaries between known and unknown regions of the occupancy grid. Discard frontiers that are unreachable. If the currently selected frontier has disappeared or becomes unreachable due to obstruction by another robot, randomly select a new frontier.

At the beginning of a deployment, the leader robot leads the sensor-limited robots in a chain formation to the vicinity of the goal destination of the first simple robot. During the navigation mode, the simple robots use a crude camera and a color blob tracking algorithm to follow the robot ahead of it, which is outfitted with a rectangular red blob. Each robot in the chain follows the robot directly in front of it at an average deployment speed of approximately 0.5 m/s. If the blob is lost, the simple robot tries to reacquire the blob by continuing its previous action or by panning itself from side to side. The blob tracking results in a follow-the-leader chaining behavior if multiple robots are front-to-back with each other. Once the leader reaches the vicinity of the sensor net position, it autonomously navigates the first simple robot into deployment position by using the camera mounted on the leader. The leader maintains the chain and proceeds to the next deployment position to provide the similar navigational assistance for deployment. Therefore, the leader visits all deployment positions in turn and deploys single sensor robots one by one.

To enable the leader robot to determine the position of any simple robot, a visual fiducial is mounted on each simple robot. Each visual fiducial consists of a start/end block, a 7-bit ID block, and an orientation block. Once the simple robots are in position, they switch state to their primary role of forming a distributed acoustic sensor network for intruder detection. They are utilized to detect acoustic targets that are moving through the environment.

9.12 COORDINATED ACTUATOR MOVEMENT FOR ENERGY-EFFICIENT SENSOR REPORTING

Mobility of robots is utilized to save energy in long-term communication tasks (e.g., video monitoring) in Liu *et al.* (2007). In such long-term communication tasks, the traffic will be regular and large enough in volume to warrant nodes expending energy moving in order to forward traffic in a more energy-efficient manner. Given a communication request between a source-destination pair, the problem is to find an initial route between them if possible (using current robot locations), and move each node on the route to its desired final location (while maintaining the route if possible). The objective is to minimize the total transmission power of nodes for this long-term communication, while keeping low movement distances from the initial to the final positions of intermediate nodes (robots).

Suppose a source S needs to find a route to a destination D. According to Goldenberg *et al.* (2004), the optimal locations of relay nodes must lie evenly on the line between S and D. Theorem 9.1 shows the optimal number of hops and optimal distance of adjacent nodes on the line. Assume that the energy needed for transmitting and receiving between two nodes at distance d is proportional to $d^\alpha + c$ for some constant α (between 2 and 6) and $c > 0$. Let $d(S, D)$ be the Euclidean distance between S and D.

Theorem 9.1. Stojmenovic and Lin (2001) Total transmission power of route from S to D is minimal when the optimal number of hops on the route is integer k, such that $|k - d(S, D) \times ((\alpha - 1)/c)^{1/\alpha}|$ is minimized, and the optimal distance of adjacent nodes is *d(S, D)/k.*

Proof. Suppose that an arbitrary path is initially established, with k hops. Its total length is no less than $|SD|$. Therefore, nodes on the path can be moved along the line SD without increasing the length of any hop, therefore reducing energy consumption on each hop. Consider now, two consecutive hops with total length t, with hop lengths $t - x$ and x, respectively. The total energy for these two hops is $(t - x)^\alpha + c + x^\alpha + c$. The first derivative of this function has zero at $x = t/2$. This means that the energy is minimal when retransmitting nodes are at equal distances to the previous and the next nodes on the path. Therefore, all relay nodes must be evenly spaced along the line, for minimal overall energy consumption of the route. The distance of adjacent nodes is then *d(S, D)/k*. So the total transmission power of nodes on the line is then $k \times ((d(S, D)/k)^\alpha + c)$. Let $x = d(S, D)/k$ denote the distance of adjacent nodes on the line. Then the total energy is $d(S, D) \times (x^\alpha + c)/x$. The first derivative of this function has zero at the value of x that minimizes the total transmission power of nodes on the line. Therefore $d(S, D) \times (\alpha - 1)x^{\alpha-2} - d(S, D) \times c \times x^{-2} = 0$. Thus, we get $x = (c/(\alpha - 1))^{1/\alpha}$. The optimal number of hops, k, can be computed by rounding $d(S, D) \times ((\alpha - 1)/c)^{1/\alpha}$ to the nearest integer. The optimal space between adjacent nodes on the line is then *d(S, D)/k*, which is a rounded value of $(c/(\alpha - 1))^{1/\alpha}$. ♦

On the basis of Theorem 9.1, two routing algorithms were proposed in Liu *et al.* (2007) to find an initial path from the source to the destination. The first algorithm is referred to as *optimal hop count routing* (*OHCR*). The current node selects a neighbor such that the neighbor is closest to the location that makes optimal progress toward the destination. Only neighbors closer to the destination than the current node are considered. Route failure will be reported to the source from the current node if no advance is possible. Source S rounds $d(S, D) \times ((\alpha - 1)/c)^{1/\alpha}$ to the nearest integer k, and computes the optimal distance of adjacent nodes $d(S, D)/k$. If $k \leq 1$ and $d(S, D) \leq r$, S transmits directly to D. Otherwise, S starts a route discovery process. Each current node u on the route reports *Route Failure* to S if there is no neighbor closer to the destination than u. Otherwise, u selects an advancing neighbor v such that $|d(v, D) - |d(u, D) - d(S, D)/k||$ is minimized. The route progresses hop by hop until destination D is reached or route failure is found. The detailed algorithm is formally presented as follows.

Optimal Hop Count Routing (OHCR) (Liu *et al.*, 2007)
Input: locations of S and D, transmission range r.

```
Output: a route from $S$ to $D$.
Begin
round $d(S, D)/min((c/(\alpha - 1))^{1/\alpha}, r)$ to the nearest integer $k$;
compute optimal distance of adjacent nodes $d(S,D)/k$;
if $k \leq 0$ and $d(S,D) \leq r$ then
    $S$ transmits directly to $D$ else $u=S;$
while $D$ is not reached && Route Failure is not found do {
    // for current node $u$ $(u \neq D)$.
    if $\{v \mid d(u,v) \leq r,\ d(v,D) \leq d(u,D)\} = \Phi$ then
        $u$ reports Route Failure to $S$
    else {
        $u$ selects neighbor $v$ such that $|d(v,D) - |d(u,D) - d(S,D)/k||$
                is minimized;
        $u$ transmits to $v$ a Route Discovery packet attached with $d(S,D)/k;\ u=v$
            } }
End
```

Once D is reached, an initial route from S to D is found. All nodes on the route are required to be moved to their optimal locations. Two moving strategies, called *move in rounds* and *move directly* were studied in Liu *et al.* (2007). Move in rounds strategy is used in algorithms proposed in Goldenberg *et al.* (2004). In each round, each node (except for S and D) moves to the midpoint of its upstream node and downstream node on the route. It requires synchronous rounds by all nodes to preserve connectivity while moving. If connectivity is not required during the movement, or could be guaranteed while moving by the topology, the unnecessary zigzag movement of nodes on the route could be replaced by the move directly strategy, where nodes move directly to their optimal locations.

Routing to D may contain different number of hops instead of the planned number k, and intermediate nodes need to learn the exact number so that they can move toward their proper final positions. D computes the actual distance of adjacent nodes on the path, and attaches the information to a *Route Confirmation* packet that is routed backwards to all nodes on the route. Upon receiving a *Route Confirmation* packet (except for S and D), each node computes its desirable location according to the attached actual hop count. The same packet also synchronizes nodes so that then can start moving toward their final positions at the same time.

A variant of OHCR, referred to as *dynamic optimal progress routing* (*DOPR*) (Falcon *et al.*, 2009), is to adjust optimal progress toward destination on the basis of the actual progress already made and the remaining number of nodes in future optimal route. The algorithm is as follows.

Dynamic Optimal Progress Routing (DOPR) (Falcon *et al.*, 2009)
Input: locations of S and D, transmission range r.
Output: a route from S to D.

```
Begin
round $d(S, D)/min((c/(\alpha - 1))^{1/\alpha}, r)$ to the nearest integer $k$;
$p = 0$; $u=S$;
while $D$ is not reached && Route Failure is not reported do {
// for current node $u$ ($u \neq D$).
   if $\{v \mid d(u,v) \leq r,\ d(v,D) \leq d(u,D)\} = \Phi$
      $u$ reports Route Failure to $S$;
else if $p \geq k$
      $u$ selects neighbor $v$ which is closest to $D$;
      $u \leftarrow v$;
else // ($p<k$)
      $DoP = d(u,D)/(k-p)$; // ideal distance for remaining nodes on the path
      $u$ selects neighbor $v$ such that $|dist(v, D) - |dist(u, D) - DoP||$ is minimized;
      $p=p+1$; $u \leftarrow v$; }
End
```

Here, p is the number of relay nodes already in the route while $k - p$ stands for the optimal number of remaining nodes to be added to the path. Hence, *DoP* is the dynamic optimal progress. When $p \geq k$, DOPR behaves as the greedy routing to attain the greatest progress to destination.

The *minimum power over progress routing* (*MPoPR*) (Liu *et al.*, 2007) is to minimize transmission power of unit progress in selecting a forwarding neighbor. The algorithm has the same structure as the previous two, and the main difference is the criterion used to select the best forwarding neighbor. Here, u selects neighbor node v, such that $(d(u, v)^{\alpha} + c)/(d(u, D) - d(v, D))$ is minimized. Note that $d(u, v)^{\alpha} + c$ is the transmission power for selecting v as a forwarding neighbor, and $d(u, D) - d(v, D)$ is the distance progress by node v. The metric denotes transmission power per unit progress. It is one of the cost over progress paradigms from Stojmenovic (2006).

In previous algorithms, only neighbors closer to the destination than the current nodes are considered. If such neighbors do not exist, route failure will be reported. The algorithms are therefore practical only for dense networks. In sparse networks, an initial route may not be found even if there exists such a route. To deal with sparse networks, recovery schemes are integrated into routing in Falcon *et al.* (2009). The aim is to employ DFS (Vukojevic *et al.*, 2008) as the built-in failure recovery mechanism for routing protocols (see also Chapter 4). The general idea is as follows.

In DFS-based routing, each node memorizes if it has already been *visited* by the DFS traversal, and the *sender* from where the message was received for the first time. It also overhears transmissions from neighbors to learn about their possible visiting status. The node that is currently holding the routing message will sort all its *unvisited* neighbors by using OHCR criterion. The first node in the list is selected to forward the message. If a node has no unvisited neighbors to proceed, it returns the message to the *sender*, which will forward the message to the next *unvisited* neighbor in the list. Eventually, the message either reaches the destination, or returns back to the source, which has no unvisited/unexplored neighbors. In the former case, a route from the source to the destination is found. The latter case indicates that the source and the destination are disconnected. The OHCR-DFS that integrates OHCR with DFS operates as follows.

```
**Optimal Hop Count Routing with Depth First Search (OHCR-DFS)** (Falcon et al., 2009)
Input: locations of $S$ and $D$, transmission radius $r$.
Output: a route from $S$ to $D$.
Begin
$S$ is marked as visited; $disconnected\_flag=0$;
$A=S$; // $A$ is the node that currently holds the message.
while $D$ is not reached && $disconnected\_flag=0$ do {
        if $A=S$ && $S$ has no unvisited neighbors
         $disconnected\_flag=1$; // $S$ and $D$ are disconnected.
        if $A \neq S$ && $A$ has no unvisited neighbors
        return the message to its sender; $A=$ sender to $A$;
        if $A$ has unvisited neighbors
        $A$ sorts all unvisited neighbors using OHCR criterion;
        $A$ sends the message to the first neighbor $B$ in the list;
        $B$ memorizes $A$ as the sender;
        $B$ is marked as visited;
        $A=B$; }
if $disconnected\_flag=1$
        report Route Failure; //disconnected topology.
else
        output the route from $S$ to $D$;
End
```

In a similar manner, DOPR can be extended to DOPR-DFS. Note that OHCR-DFS and DOPR-DFS output the same routes as OHCR and DOPR if OHCR and DOPR succeed. However, OHCR-DFS and DOPR-DFS can cope with sparse networks and are guaranteed to find routes as long as the source and the destination are connected.

In the example in Figure 9.9, S initiates routing to D. First, S marks itself as visited, and sorts neighbors A and F according to OHCR criteria. Suppose A is the first node in the list, that is, $|d(A, D) - |d(S, D) - d(S, D)/k|| < |d(F, D) - |d(S, D) - d(S, D)/k||$, where k is the optimal hop count, in terms of energy efficiency. S forwards the message to A, which then forwards to B. B sorts neighbors C and E in a similar way as S does, and forwards the message to C. C then forwards to E. Until this step, OHCR-DFS operates exactly the same as OHCR does. Since E has no unvisited neighbors, OHCR will report Route Failure and routing terminates here. In OHCR-DFS, however, E returns the message to its sender, that is, C, which then returns to B. Note that B overhears the visited status of E when E returns the message to C via broadcasting. Thus, B forwards the message to its only unvisited neighbor G. The message eventually reaches

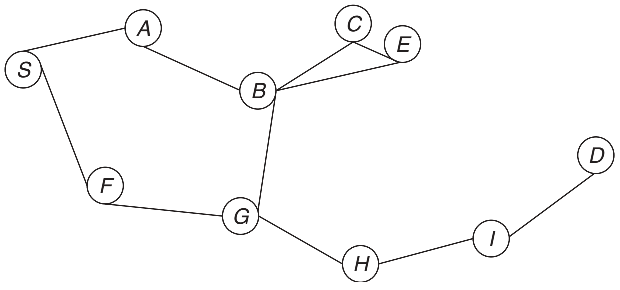


Figure 9.9 Routing of OHCR-DFS.

D via H and I. Therefore, the trajectory of routing message is *SABCECBGHID* and the route from S to D is *SABGHID*.

Collect K neighbors routing (CKNR) was further proposed in Falcon *et al.* (2009) to deal with the networks where the source and the destination may even be disconnected. Once the optimal hop count k, is computed, one can simply run DFS over the entire graph until it finds exactly k nodes. All k nodes are afterwards moved directly to their ideal locations. Note that it could be possible that the destination is reached before k nodes are collected. In this case, the routing algorithm terminates.

Depending on the criterion used in selecting neighbors, we have CKNR-OHCR and CKNR-DOPR. In CKNR-OHCR, a forwarding node is selected as in OHCR-DFS. The only difference is the termination condition. CKNR-OHCR terminates once k nodes are collected or the destination is reached while OHCR-DFS terminates when the destination is reached. Therefore, even if the source and the destination are disconnected, CKNR-OHCR can find k nodes, which move to their desired locations to construct a feasible route. Note that CKNR-OHCR may select nodes that have no sequential neighboring relationship. Each node is assigned with a sequential number (counter), which indicates the place of the node in the final route after movement.

In the example in Figure 9.10, suppose the optimal hop count is $k = 6$. S first marks itself visited and is assigned 1. S sorts neighbors A and F according to OHCR criterion, and forwards the message to A, which is assigned 2.

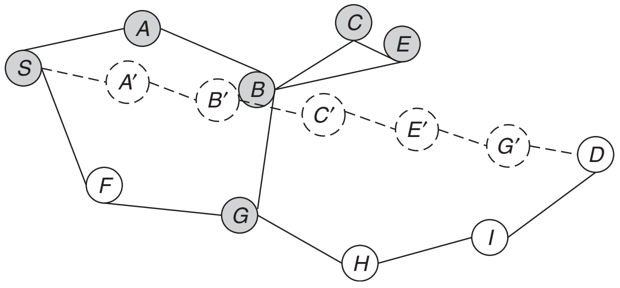


Figure 9.10 Routing of CKNR-OHCR.

A forwards the message to B, which is assigned 3. B sorts neighbors C and E, and forwards the message to C. C then forwards to E. C and E are assigned 4 and 5, respectively. Since E has no unvisited neighbors, E returns the message to its sender, that is, C, which then returns to B. The counter does not increase until B forwards the message to its only unvisited neighbor G, which is assigned 6. The routing algorithm terminates, as six nodes have been collected. After G replies an ACK message to S along the path, each node knows the actual number of nodes in the route. Each node computes its desired location according to its sequential number and moves directly to the location. The final route from S to D is $SA'B'C'E'G'D$ in Figure 9.10.

9.13 FLYING ROBOTS

The team from the Berlin Technical University is working on miniaturized robot helicopters with advanced intelligence (Dumiak, 2009). The project is called *Autonomous Flying Robot MARVIN* (*multipurpose aerial robot vehicles with intelligent navigation*) (MARVIN). Unlike other robocopters studied by several groups around the world, these flying robots (MARVIN) are able to coordinate with each other to complete required tasks. The main applications include (i) load transport with multiple helicopters; (ii) deployment of sensor networks using small scale aerial robots; and (iii) monitoring and observation. Robocopters can be expected for use to distribute sensors that would help coordinate firefighting efforts or search flood zones, to track or find people and vehicles, or to record movies and cover sports events (Dumiak, 2009). Hoisting communications gear, they can even provide services of WiFi or mobile phone traffic if infrastructure has been destroyed by an earthquake or some other natural disaster.

There are three main types of robots in the MARVIN project: helicopters, quad-rotors, and planes. The specification of above three robots is listed in Table 9.1.

The autonomous navigation of these flying robots (MARVIN) is performed by a modular control system. The control system can be used to operate different types of flying robots. The modular control system is composed of hardware for autonomous navigation, real-time software for control and communication as well as of mission software. The control system can be configured for different types of actuators and sensors, for example, low-cost and high-end navigation

Table 9.1 Specification of Robocopters (MARVIN).

	Size (m)	**Weight (kg)**
Helicopters H-3	Rotor diameter 2	12–16
Quad-rotor H-Q1	Edge length 1	15
Plane H-P1	Wingspan 2.5	15

sensors. The copters' control systems allow small craft to work together in lifting loads and scouting environments. For instance, suppose there are three or four copters each of which shares the load while tethered by a rope to a single object. Such operation causes many contrary forces for the copters. If those copters are manually controlled, it is not easy to make a stable operation. However, the autonomous robocopters are able to make instant and coordinated adjustments and keep the equilibrium. Each robot accounts for the location of the other helicopters, the forces coming from them, and the load on the rope, to jointly lift an object. The robocopters' coordination system integrates four software modules for stabilizing the copter: navigation; exploration; obstacle avoidance; and processing orientation, horizon, and position. The robocopters can also be used for more efficient scouting because they automatically divide an area among themselves (Dumiak, 2009). Robocopters are also used to deploy distributed communications networks by dropping off a batch of tiny sensor nodes. Each sensor node is equipped with a data processor, a radio, a battery, and sensors, which could detect and measure temperature, light, radiation, location, and chemical values. The sensor's transmission range is about 25 m. Thousands of these sensor nodes could be distributed by robocopters to build a self-organized network, which is used to survey a forest fire or flood zone for rescue efforts. An ongoing plan of the MARVIN project is to use these robocopters to deploy a sensor network to track mobile objects and people, follow movement inside and outside of buildings, and monitor with high-end airborne cameras.

REFERENCES

Akyildiz IF, Kasimoglu IH. "Wireless sensor and actor networks: research challenges". Ad Hoc Netw 2004;2(4):351–367.

Bekey G, Yuh J. "The status of robotics". IEEE Rob Autom Mag 2008;15(1):80–86.

Correll N, Cianci C, Raemy X, Martinoli A. "Self-organized embedded sensor/actuator networks for "smart" turbines". IROS 2006 Workshop: Network Robot System: Toward Intelligent Robotic Systems Integrated with Environments; Beijing, China; 2006 Oct.

Correll N, Martinoli A. "Robust distributed coverage using a swarm of miniature robots". Proceedings of IEEE Conference on Robotics and Automation; Roma, Italy; 2007a. pp. 379–384.

Correll N, Martinoli A. "Modeling self-organized aggregation in a swarm of miniature robots". Proceedings of IEEE 2007 International Conference on Robotics and Robotics and Automation Workshop on Collective Behaviors Inspired by Biological and Biochemical Systems; Roma, Italy; 2007b.

Dudek G, Jenkin M, Milios E. "A taxonomy of multirobot systems" . In: Balch T, Parker LE, editors. Robot teams: from diversity to polymorphism: A.K. Peters; 2002. pp. 3–22.

Dumiak M. "Robocopters Unite!" IEEE Spectrum Online. Available at http://www.spectrum.ieee.org/feb09/7368. 2009 Feb.

Falcon R, Liu H, Nayak A, Stojmenovic I. "Power-aware routing for localized controlled mobility in robotic wireless sensor networks". 2009. In preparation.

Farinelli A, Iocchi L, Nardi D. "Multirobot systems: a classification focused on coordination". IEEE Trans Syst Man Cybern Part B 2004;34(5):2015–2028.

Gerkey BP, Mataric MJ. "A formal analysis and taxonomy of task allocation in multi-robot systems". Int J Robot Res 2004;23(9):939–954.

Goldenberg DK, Lin J, Morse AS, Rosen BE, Yang YR. "Towards mobility as a network control primitive". Mobihoc'04; Japan; 2004. pp. 163–174.

Howard A, Parker LE, Sukhatme G. "Experiments with a large heterogeneous mobile robot team: exploration, mapping, deployment, and detection". Int J Robot Res 2006;25(5–6):431–447.

Jeanson R, Rivault C, Deneubourg J-L, Blanco S, Fournier R, Jost C, Theraulaz G. "Self-organized aggregation in cockroaches". Anim Behav 2004;69:169–180.

Kumar V, Rus D, Sukhatme GS. "Networked robots". In: Bruno S, Oussama K, editors. Handbook of robotics: Springer; 2008. pp. 943–958.

Li H, Yu HY, Yang BW, Liu A. "Timing control for delay-constrained data aggregation in wireless sensor networks". Int J Commun Syst 2007;20:875–887.

Liu H, Nayak A, Stojmenovic I. "Localized mobility control routing in robotic sensor wireless networks". Proceedings of the 3rd International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2007), LNCS 4864; Beijing, China; 2007. pp. 19–31.

MARVIN. Multi-purpose aerial robot vehicles with intelligent navigation. TU Berlin: Department of Computer Science. Available at http://pdv.cs.tu-berlin.de/MARVIN/.

Melodia T, Pompili D, Gungor VC, Akyildiz IF. "Communication and coordination in wireless sensor and actor networks". IEEE Trans Mobile Comput 2007;6(10):1116–1129.

Mezei I, Malbasa V, Stojmenovic I. "Communication aspects of robot-robot coordination for single task single robot wireless multi-hop networks". IEEE Rob Autom Mag, to appear.

McLurkin J, Smith J. "Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots". Proceedings of Distributed Autonomous Robotic Systems; Toulouse, France; 2004 Jun.

McLurkin J, Yamins D. "Dynamic task assignment in robot swarm". Proceedings Robotics: Science and Systems; Cambridge, Massachusetts; 2005 Aug. pp. 129–136.

Ngai ECH, Lyu MR, Liu J. "A real-time communication framework for wireless sensor-actuator networks". Proceedings of 2006 IEEE Aerospace Conference. Montana: Big Sky; 2006.

Ogren P, Fiorelli E, Leonard NE. "Formations with a mission". Proceedings MTNS; Notre Dame, France; 2002.

Ozaki K, Hayashibara N, Enokido T, Takizawa M. "Fault tolerant semi-passive coordination protocol for a multi-actuator/multi-sensor (MAMS) model". Proceedings of IEEE International Conference Availability, Reliability and Security ARES; Austria; 2007.

Parker LE. "Multiple mobile robot systems". In: Bruno S, Oussama K, editors. Handbook of robotics: Springer; 2008. pp. 921–941.

Seah WKG, Liu Z, Lim JG, Rao SV, Ang MHJr. "TARANTULAS: mobility-enhanced wireless sensor-actuator networks". Proceedings of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing; Taichung, Taiwan; 2006. pp. 548–551.

Selvaradjou K, Dhanaraj M, Murthy CSR. "Energy efficient assignment of events in wireless sensor and mobile actor networks". Proceedings of 14th IEEE International Conference on Networks; Singapore, Volume 2; 2006. pp. 1–6.

Shah K, Meng Y. "Communication-efficient dynamic task scheduling for heterogeneous multi-robot systems". Proceedings of International Symposium on Computational Intelligence in Robotics and Automation; Jacksonville, USA; 2007. pp. 230–235.

Stojmenovic I. "Localized network layer protocols in sensor networks based on optimizing cost over progress ratio". IEEE Netw 2006;20(1):21–27.

Stojmenovic I, Lin X. "Power aware localized routing in Ad hoc networks". IEEE Trans Parallel Distrib Syst 2001;12(10):1023–1032.

Vukojevic B, Goel N, Kalaichevan K, Nayak A, Stojmenovic I. "Depth first search based and power aware geo-routing in ad hoc and sensor wireless networks". Int J Auton Adapt Commun Syst IJAACS 2008;1(1):41–54.

Wong SY, Lim JG, Rao SV, Seah WKG. “Multihop localization with density and path length awareness in non-uniform wireless sensor networks”. Proceedings of IEEE VTC 2005 (Spring); Dallas, USA; Volume 4; 2005. pp. 2551–2555.

Yamauchi B. “Frontier-based approach for autonomous exploration”. Proceedings of the IEEE International Symposium on Computational Intelligence, Robotics and Automation; Monterey, USA; 1997. pp. 146–151.

Chapter 10

Sensor Placement in Sensor and Actuator Networks

Xu Li [1], **Amiya Nayak**[1], **David Simplot-Ryl**[2], **and Ivan Stojmenovic**[1]

[1] *School of Information Technology and Engineering, University of Ottawa, Ontario, Canada K1N 6N5*
[2] *CNRS/INRIA, University of Lille 1, 59655 Villeneuve d'ascq, France*

Abstract

Coverage is a functional basis of any sensor network. The impact on coverage from stochastic node dropping and inevitable node failure, coupling with controlled node mobility, gives rise to the problem of *movement-assisted sensor placement* in wireless sensor and actuator networks (WSAN). One or more actuators may carry sensors, and drop them at proper position, while moving around, in the region of interest (ROI) to construct desired coverage. Mobile sensors may change their original placement so as to improve existing coverage. Emerging coverage holes are to be covered by idle sensors. Actuator may place spare sensors according to certain energy optimality criteria. If sensors are mobile, they can relocate themselves to fill holes. In this chapter existing solutions to the sensor placement problem in WSAN are comprehensively reviewed.

10.1 INTRODUCTION

Sensor networks aim at monitoring their surroundings for event detection and/or object tracking (Akyildiz *et al.*, 2002; Martincic *et al.*, 2005). Because of this surveillance goal, *coverage* is a functional basis of any sensor network. In order to best fulfill its designated surveillance tasks, a sensor network must maximally or fully cover the right region, where interesting events occur, without internal

Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication. By Amiya Nayak and Ivan Stojmenovic. Copyright © 2010 John Wiley & Sons, Inc.

sensing holes. Sometimes, additional requirements such as node degree (Poduri *et al.*, 2009), node density (Garetto *et al.*, 2007), or coverage focus (Garetto *et al.*, 2007; Li *et al.*, 2008a) may apply.

However, it cannot be expected that sensors are placed in a desired way at initiation as they are often randomly dropped due to operational factors. Furthermore, sensors could fail at runtime for various reasons such as power depletion, hardware defects, and damaging events, degrading already poor coverage. In WSAN, the impact on coverage from stochastic node dropping and unpredictable node failure, coupling with controlled node mobility, brings about the problem of *movement-assisted sensor placement* for coverage formation and improvement.

There are different ways to place sensors by exploiting node mobility in WSAN. Sensors can be placed by mobile actuators. If sensors have locomotion, then they can place themselves by intelligently changing their geographic location without others' help. Because physical movement (including starting motors) consumes a large amount of energy, a movement-assisted sensor placement scheme is expected to yield a small number of moves and small total moving distance. As sensors are often dropped in an unknown environment, terrain and boundary information of the sensory field may not be known a priori, and the algorithm is expected to enable mobile sensors/actuators to avoid physical obstacles on the fly.

So far, a number of movement-assisted sensor placement algorithms have been proposed. But, nevertheless, no systematic study on these algorithms has yet been presented. This chapter will fill this gap. Section 10.2 introduces four subtopics of the movement-assisted sensor placement problem. Section 10.3 presents basic technique for sensor migration between two points. Sections 10.4–10.7 survey the major research efforts on these topics in detail. We will denote by r_s sensing radius and by r_c communication radius.

10.2 MOVEMENT-ASSISTED SENSOR PLACEMENT

Four subproblems of movement-assisted sensor placement have been investigated for coverage improvement in the literature. This section gives these problems a general definition. A comprehensive survey of existing solutions can be found, respectively, in the following sections.

10.2.1 Sensor Placement by Actuators

Actuators may serve as network *installers* for sensor deployment. They carry sensors as payload and move around in the ROI. While traveling, they deploy sensors at desired positions (e.g., vertices of certain geographic graph) to "install" a connected sensor network with desired coverage.

If ROI is bounded, and there are sufficient sensors, the key problem is how to guide actuators to explore entire ROI. Otherwise, the challenge will be how to ensure a coverage of good *compactness*. Compactness can be measured by the radius of the maximum hole-free disk in the final network. It reflects the omni-sensibility of the network.

10.2.2 Coverage Maintenance by Actuators

After initial sensor deployment, actuators can be used as network *maintainer* to improve existing coverage by planting sensors at designated location. Specifically, upon request, they will move to reported sensing holes (e.g., due to improper initial node distribution or runtime node failure) and drop new sensors there to fill the holes with minimum delay. If actuators have no sensors in hand, then they have to first fetch spare sensors in the network. The delay from the moment when an actuator received a request to the moment when it filled the reported hole ought to be minimized.

Consider a single actuator case with a redundant sensor at any point in covered areas and every sensing hole is small enough to be patched by a single sensor. In this hand-made special scenario, the actuator's only task is to find a shortest tour that visits every sensing hole exactly once, which is actually the NP-complete traveling salesman problem. Hence, the actuator-based coverage maintenance problem is NP-hard.

10.2.3 Sensor Self-Deployment

Sensor self-deployment takes place immediately after initial sensor dropping. It aims at achieving desired coverage through network-wide autonomous sensor reorganization. To perform self-deployment, each sensor node needs to have locomotion.

Sensor self-deployment was first introduced by Howard *et al.* (2002a). It is closely related to robot exploration and mapping problem (Lopez-Sanchez *et al.*, 1998; Yamaguchi *et al.*, 1998; Burgard *et al.*, 2000) and pattern formation problem (Sugihara and Suzuki, 1996; Scheider *et al.*, (2000); Fredslund and Mataric, 2001) in the field of mobile robots. But, it differs from these problems in model definition: mobile sensors have "hearing", while mobile robots have "vision".

10.2.4 Sensor Relocation

Sensor relocation deals with failure nodes within a sensor network, that is, how to replace emerging failed sensors with redundant ones through nodal geographic migration, without topological change. To perform sensor relocation, each sensor node is required to have locomotion. Sensor relocation involves two tasks: *replacement discovery*, that is, finding a redundant sensor as the replacement of a failed node, and *replacement migration*, that is, migrating the replacement to the failed sensor's position.

10.3 MOBILE SENSOR MIGRATION

After a mobile sensor makes its decision for self-deployment or relocation, it will migrate from its current position to the target position. Mobile sensor migration

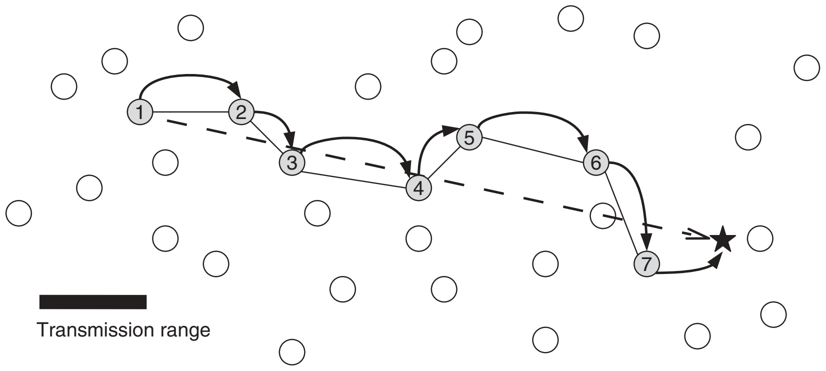


Figure 10.1 Shifted migration method.

can be accomplished in a *direct* way or in a *shifted* manner. In *direct migration*, the sensor simply moves all the way to the target location. Owing to the potentially long moving distance, this method can cause long migration delay and large energy consumption on the migrating sensor. In *shifted migration*, a multi-hop migration path is built from the sensor to the target location. As illustrated in Figure 10.1, every sensor along this path shifts its position by one-hop toward the target location. The last sensor in the path moves to the target location.

Compared with direct migration, shifted migration may generate longer total moving distance and a larger number of moves, because migration path is usually not short and often composed of multiple nodes. However, instead of punishing only one node, this method distributes energy consumption among all the nodes along the path, prolonging network lifetime as a whole. In addition, it renders migration latency proportional to the longest hop (not longer than the communication radius r_c) rather than the Euclidean distance between the target location and the sensor. In these cases, shifted migration is more desirable than direct migration. The key to shifted migration is energy-efficient migration path discovery, which is in fact a routing problem (described in Chapter 4). In order not to jeopardize the execution of other network protocols, every shifting node must transfer all its local data to the replacement node at its original position after the shifted migration process.

10.4 SENSOR PLACEMENT BY ACTUATORS

Till the time of this writing, only a few algorithms (Batalin and Sukhatme, 2005; Chang *et al.*, 2007; Fletcher *et al*., 2009) were proposed to address how to deploy sensors by actuators. In this section these algorithms are reviewed in detail.

10.4.1 Least Recently Visited Approach

Batalin and Sukhatme (2005) presented a single-actuator-based sensor placement algorithm LRV (least recently visited), which assumes equal sensing and communication radii and guides actuator movement according to the suggestion of previously deployed sensors. The algorithm starts with an empty environment. At initiation, the actuator (robot) deploys a node at its current position. Each deployed sensor maintains a set of directions along which the robot can move away from it. Directions could follow a graph structure (e.g., tree) or could be predefined (e.g., four geographical directions). It also assigns a weight, initially equal to 0, to each direction, indicating the number of times that direction was traversed by the actuator.

Every sensor recommends its locally LRV direction to the actuator by message when the actuator is in its communication range. Directions are preordered so that a single direction is recommended in case of a tie. The actuator travels a predefined distance in recommended direction. If, however, the chosen direction is obstructed, it will inform the recommender and ask for a new suggested direction. Whenever the actuator departs or arrives, its current sensor increases the weight of its going direction (respectively, coming direction). This can be done by the sensor upon the actuator's notification. The actuator remains at a location for a predefined short period before its next movement. During this period, if it receives no sensor message, it will drop a new sensor in the environment.

Figure 10.2 is an illustration of the LRV approach. A robot starts from location A and travels in four geographic directions ordered as south, east, north, and west. In the figure, the thick arrow indicates robot trajectory; numbers around a node imply local weight of the four directions. Crosses are used to mark directions that are locally known (from robot's information) to be obstructed. The robot drops at location A a sensor, which then suggests it to move to the south. Following the suggestion, the robot moves to location B, drops a new sensor there, and takes it as the current sensor. The current sensor first recommends direction south (which is found by the robot obstructed) and then the east to the robot. The robot proceeds this way and reaches location C, as shown in Figure 10.2a. Then, it has to travel to its incoming direction, north, because all the other directions are obstructed. It keeps traveling according to the previously deployed sensors' recommendation dropping sensors at proper positions, and finally returns to the starting point A, as shown in Figure 10.2b.

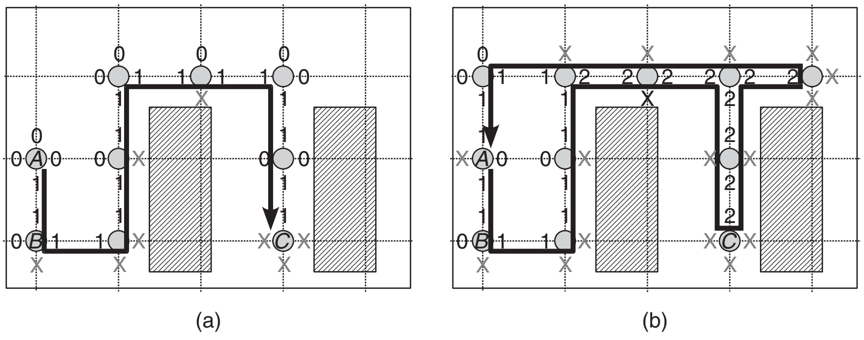


Figure 10.2 Least recently visited approach: (a) Snapshot1 and (b) Snapshot2.

LRV is a purely localized algorithm and thus message efficient and fault tolerant. Although the authors proved that the exploration time of LRV on a finite graph is finite, it is not clear under what conditions the algorithm terminates. Because the robot has no global view about the coverage and always receives a recommended direction from its current sensor, it will not stop moving unless it has no sensor left in hand. However, the exhaustive movement enables the robot to visit and fill (by dropping new sensors) emerging sensing holes caused by runtime node failure (with low path efficiency for discovering such search due to random nature of recommendations in areas far from holes).

10.4.2 Snake-Like Deployment Approach

Chang *et al.* (2007) presented a snake-like sensor placement approach, referred here to as SLD (snake-like deployment). SLD uses a single mobile actuator to deploy static sensors at vertices of an equilateral triangle tessellation (TT) constructed over a bounded rectilinear ROI. The single actuator moves like a snake, starting from the upper-left corner of the ROI. It moves to the right along a horizontal line and drops sensors at separation $\sqrt{3}r_s$ until it hits the boundary of the ROI or an obstacle. Then it moves a distance of $\frac{\sqrt{3}}{2}r_s$ down to the next horizontal line, changes its moving direction to the left, and proceeds similarly.

The algorithm also attempts to avoid sensing holes hidden behind physical obstacles by allowing the actuator to break its regular movement pattern. Specifically, the actuator checks, before its next movement step, whether there is any sensing hole in its vicinity in its coming direction. If the answer is positive, it will change its moving direction toward that hole. By this means, the actuator can move up and down, left and right, along different lines, reducing the occurrence possibility of sensing holes. It also allows the actuator to start not only from a corner of, but also from the middle of, the ROI.

However, this algorithm remains incomplete. It is not clear how the algorithm terminates, that is, under what conditions the robot stops moving. There are even unexplainable actuator behaviors in the examples used in Chang *et al.* (2007). Unlike what the authors claimed, the algorithm in fact does not guarantee full coverage according to their current algorithm description. A simple counterexample scenario is that a wall partially divides the ROI with one end attaching the border of the field, as shown in Figure 10.3. In such a situation, once the robot enters one side of the wall, it will not be able to enter the other side.

10.4.3 Back-Tracking-Deployment Approach

Fletcher *et al*. (2009) presented a localized backtracking-based sensor deployment (BTD) approach for a bounded ROI. Actuators carry sensors and are randomly scattered in the ROI. They know about their own location, and are able to detect physical obstacles, ROI boundaries, and early-deployed sensors. Four geographic directions, that is, north, west, south, and east, are assigned distinct ranks. Each

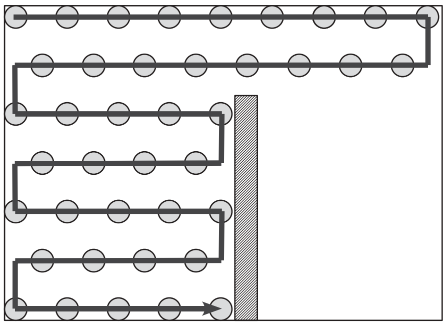


Figure 10.3 Incomplete coverage by SLD.

actuator moves independently and asynchronously toward local open direction with highest rank; if obstructed (by an obstacle, boundary of ROI, or a sensor), it chooses to move toward the direction of next highest rank. While traveling, it drops sensors at proper positions (subject to desired network topology such as square grid or TT). Unlike LRV (Batalin and Sukhatme, 2005) and SLD approach (Chang *et al.*, 2007), BTD terminates within finite time and yields full coverage over the ROI. Its details are explored below.

An *empty spot* is a geographic point where a sensor is supposed to be dropped. A sensor is said to be *adjacent* to an empty spot if it is so in the desired network topology. Each sensor is dynamically assigned a *color* according to its neighborhood status by the following rule: it is colored “white” if there exists an adjacent empty spot, otherwise “black”. The successor (or predecessor) of a sensor is the sensor that is dropped immediately after (respectively, before) it by the same actuator. Each sensor stores a *forward pointer* and a *back pointer*. The former points to the location of the sensor’s successor. The later points to the location of the sensor’s predecessor if the predecessor is white, or to the location that its predecessor’s back pointer points to otherwise. In other words, it points to the first white sensor along the backward path of its dropping actuator. If a sensor does not have a successor (or predecessor), then its forward (respectively, back) pointer is set to *nil*. Forward- and back pointers together serve as navigation tool and allow actuators to backtrack each other’s trajectory.

Actuators have distinct IDs. They inform each of their dropped sensors about their IDs and meanwhile associate it with an increasing sequence number. Sensors dropped by the same actuator have distinct sequence number, whereas those dropped by different actors may not. By a periodic HELLO message, sensors exchange with their neighbors local information such as position, dropping actuator ID, sequence number, color, and back pointer. Hence, coloring and forward/back pointer setup are both locally defined. Figure 10.4 shows the resulting configuration of these behaviors at four moments during the execution of BTD in an ROI. In this example, two actuators a and b start from different locations A and B for sensor placement with square grid topology. They move following the order of preference west > east > north > south. Their current locations are marked by small triangles; their moving destinations are indicated by thick arrow

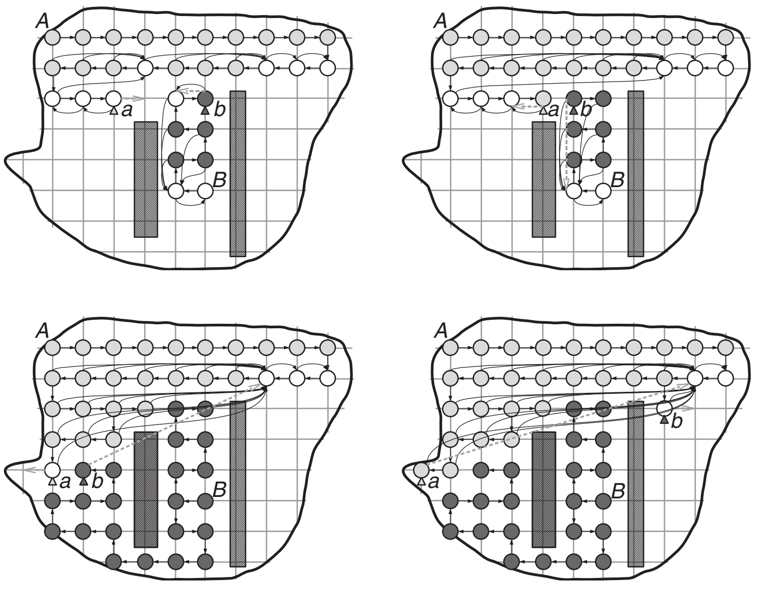


Figure 10.4 Dead ends and backtracking in BTD.

lines. Sensors' forward- and back pointers are indicated by thin straight arrow lines and curly arrow lines, respectively.

The sensor located at the current location of an actuator is called *current sensor* of the actuator. An actuator reaches a *dead end* if all the four moving directions are obstructed. In this case, the actuator backtracks along the back pointer chain starting from its current sensor to revisit previous white sensors, find their adjacent empty spots (entrances to uncovered areas), and resume ROI exploration and sensor dropping from there. If the actuator cannot find a non-nil back pointer on its current sensor, it will check whether there is one stored back pointer in its neighborhood. If the result is negative, it will terminate; otherwise, it moves along the back pointer stored at a neighboring sensor with maximum sequence number. In case of a tie, a random choice is made.

When an actuator is backtracking for a white sensor, we say the actuator is serving that sensor. If the number of serving actuator of a white sensor is equal to the number of its adjacent empty spots, then the sensor is considered *fully served*. Before an actuator starts to serve a white sensor, it sends a request message to that sensor. It takes actual serving action only after the request is granted, or if no reply is received after a number of retrials. The sensor-dropping action of an actuator can change color of a nearby white sensor; the color change will trigger the change of back pointers along the forward point chain from the sensor to its dropping actuator. Owing to network asynchrony and information propagation delay, an actuator might move to a black sensor from a dead end. In this case, the actuator will reach another dead end.

Figure 10.4 shows the four dead-end situations. In Figure 10.4a, actuator *b* reaches a dead end and decides to backtrack along the back pointer of its current sensor to a white sensor. As shown in Figure 10.4b, after it reaches the destination, the white sensor turns black because its only adjacent empty spot is filled by a sensor dropped by actuator a. Both actuators are in a dead-end situation, and they decide to move backward along their paths to the first previous white sensor. Later, actuator *b* reaches another dead end in Figure 10.4c. Because it cannot find a non-nil back pointer on its current sensor, it backtracks along the back pointer stored on a neighboring sensor dropped by actuator a. Afterwards, actuator a also reaches a dead end in Figure 10.4d and performs backtracking.

Single node failure in a back pointer chain does not affect algorithm execution. It is because an actuator will be led to the failure sensor's position and replace it with a new one, and the actuator will then recover the back pointer stored on the failure node from its predecessor. If multiple adjacent sensors fail together, a sensing hole occurs. Likewise, an actuator will be led into the hole region. It will treat the hole as uncovered area and drop sensors there. Any white sensor outside the hole region can be identified through the back pointers stored along the boundary of the hole, and the actuator will then place sensor following those pointers after the hole is patched.

10.5 COVERAGE MAINTENANCE BY ACTUATORS

It has not yet been well studied how to repair/maintain coverage using actuators. Existing solutions (Mei *et al.*, 2007) are straightforward application of clustering and flooding with huge message overhead. They work under the assumption that actuators are carrying sufficient spare sensors. However, with this assumption, we can obtain a more efficient solution by combining face routing (Bose *et al.*, 1999) and anycasting (Mitton *et al.*, 2009). Below, we first introduce the previous work (Mei *et al.*, 2007) and then describe the new solution.

10.5.1 Cluster-Based Approach

Mei *et al.* (2007) addressed how to replace failed sensors in WSAN by presenting three straightforward actuator coordination protocols. In a proposed centralized protocol, an actuator is appointed central manager and responsible for handling node failure reports. The central controller broadcasts its location to all sensors and other actuators. It maintains the latest position of each actuator by listening to actuator location updates. Sensors monitor each other and report detected node failures to the central manager, which then dispatches closest actuators to replace failed sensors with their carried spare ones. An actuator receiving multiple orders will handle them on a first-come-first-serve basis. As an actuator moves to its assigned failure location, it keeps updating the central manager with its latest position.

In a proposed distributed protocol, the sensory field is partitioned into equal-sized subregions. Each actuator is assigned one and only one subregion and required to handle regional node failure reports as manager; it is also responsible for sensor replacement in its own subregion. The centralized algorithm is then run within each subregion. In a proposed dynamic protocol, the sensory field is dynamically partitioned according to the current position of each robot. Specifically, each robot broadcasts its current location; sensors receiving messages from multiple robots rebroadcast only the one from closest robot. Finally, a Voronoi digram (to be described later in Section 10.5.2) is constructed based on hop count. Nodes report detected sensor failures to the creating actuators of their home Voronoi cells, which then move to replace the failed sensor with their carried spare ones. While moving, actuators broadcast their latest location to update the Voronoi diagram.

These three protocols are all based heavily on frequent network-wide flooding and thus very expensive in message and in energy requirements. The centralized protocol creates communication bottleneck and easily induces single-point failure. Apparently, none of these protocols is a practical candidate for large-scale sensor networks.

10.5.2 Perimeter-Based Approach

Below we propose a localized perimeter-based scheme for actuator-assisted coverage maintenance by combining anycasting and face routing. In this solution, actuators are required to form a connected network. To obtain such a network, actuators can be densely dropped in a small region first and then spread by a vector-based self-deployment approach (Section 10.5.1). Actuators locally construct a Gabriel graph (described in 4) over the actuator network.

When a sensor detects a sensing hole, which is represented by a geographic point, it sends a report to any one of the actuators by anycasting (Mitton *et al.*, 2009).

The actuator receiving the report, which is not necessarily the closest one to the reporting sensor, routes a message toward the sensing hole through greedy-face-greedy (GFG) routing protocol (Bose *et al.*, 1999) over the actuator network. A detailed description of GFG and anycasting can be found in Chapter 4. Because the routing process will fail in case of lack of destination, the message will make a cycle around that sensing hole on Gabriel graph and stop at the actuator closest to it. This actuator will take the responsibility to fill the reported sensing hole.

This scheme has obvious advantage over the algorithms in Mei *et al.* (2007) in message overhead, because it involves no flooding operation at all. The perimeter-based idea of finding a node to act on some task has been used to solve a different problem, data centric storage (Ratnasamy *et al.*, 2002).

10.6 SENSOR SELF-DEPLOYMENT

Sensor self-deployment is an active research subject that is continuously drawing large amount of attention. In the literature, it has been modeled and solved using different techniques. At the time of this writing, there exist eight different self-deployment approaches as listed below:

- *Virtual Force (Vector-Based) Approach.* Sensors move according to a movement vector computed using the relative position of their neighbors.
- *Voronoi-Based Approach.* Sensors adjust their location to reduce uncovered local area in its Voronoi polygon possibly in multiple rounds.
- *Load Balancing Approach.* The number of sensors in the regions of a partitioned sensor field is balanced through multiple rounds of scans.
- *Stochastic Approach.* Sensors spread out through random walk.
- *Point-Coverage Approach.* The area coverage problem is converted to a point-coverage problem over certain geographic graph.
- *Incremental Approach.* Sensors are deployed incrementally, that is, one at a time, based on the information gathered from previously deployed sensors.
- *Maximum-Flow Approach.* Sensors deployment is modeled as minimum-cost, maximum-flow problem from source regions to whole regions in ROI.
- *Genetic-Algorithm Approach.* Sensor movement plan is generated by multiround selection and reproduction simulating genes and nature selection.

In the above list, the first five are distributed or localized approaches. The rest are centralized approaches with requirement for a global view of the network. Their output is a motion plan for every sensor node satisfying certain optimization criteria. In the rest of this section, representative algorithms for each of the above sensor self-deployment approaches have been reviewed.

10.6.1 Virtual Force Approach

The best known sensor self-deployment approach is probably the virtual force (vector-based) approach introduced by Howard *et al.* (2002a). Thus far, many different implementations of this technique have been proposed. Although these implementations claim that they are inspired by different physical models such as potential field (Howard *et al.*, 2002a), molecules (Heo and Varshney, 2005), and electromagnetic particles (Wang *et al.*, 2004a), they really share the common philosophy at their core. That is, each node i computes virtual force (movement vectors) V_i^j due to its neighbor j using nodal relative position and moves according to the total force (vector summation) V_i; after a number of rounds of movement, the network reaches an equilibrium status, which gives a near uniform node distribution and thus a near optimal coverage.

Figure 10.5 illustrates how this basic idea works in a network of three sensors. Initially, the sensing ranges of nodes overlap (Fig. 10.5a), and virtual forces are all repulsive, leading them to move apart for coverage maximization. As the nodes move, the sensing ranges of nodes 2 and 3 tend to separate (Fig. 10.5b), and the virtual force between them turns into attractive and drives them to move

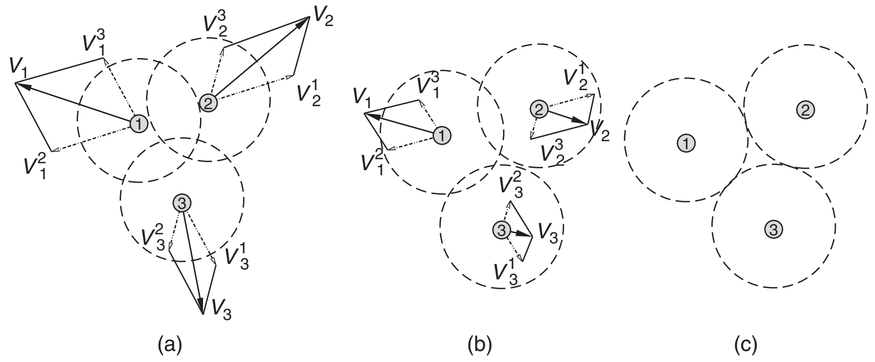


Figure 10.5 Vector-based approach: (a) initial distribution; (b) transient distribution; and (c) final distribution.

toward each other to avoid creating sensing hole. Finally, nodal sensing ranges touch each other without overlapping (Fig. 10.5c); hence, no virtual force is generated, and nodes do not move.

Existing virtual-force-based sensor self-deployment algorithms extend this basic idea by adding different terminating mechanisms or additional constraints. For example, in Howard *et al.* (2002a), virtual friction force, which is proportional to nodal velocity and always against nodal moving direction, is used to stop nodal movement so that a static equilibrium can be eventually reached. In Wang *et al.* (2004a), Voronoi diagram is used to judge nodes' coverage effectiveness and help them decide to stop moving. In Heo and Varshney (2005), node density is brought into consideration during virtual force computation for energy saving. In Ma and Yang (2007), each node receives virtual force from at most six neighbors such that the resulting network has a TT layout. Two most recent variants of this approach are described in detail below.

Poduri *et al.* (2009) studied control of positions of nodes for desired levels of network connectivity and sensing coverage. The problem is to determine positions of nodes such that the sensing coverage is maximized while satisfying the connectivity constraint. It is assumed that mobile nodes (robots) are densely deployed in the ROI. The authors proposed the neighbor-every-theta (NET) graph where each node has at least one neighbor in every θ angle sector of its communication range. That is, the angle between any two neighbors in sorted order (measured from given node) is $\leq \theta$. NET graphs are proven guaranteed to have, when $\theta < \pi$, an edge-connectivity of at least $\lfloor \frac{2\pi}{\theta} \rfloor$ even with the assumption of irregular communication range (Poduri *et al.*, 2009). A graph is said to be k-(edge) connected iff there are at least k node-disjoint paths between any two nodes in the graph. The graphs can achieve coverage-connectivity trade-offs based on a single parameter θ. If the communication range equals to the sensing range, then sensing coverage is maximized when $k \geq 3$ nodes are placed at the edges of k disjoint $\frac{2\pi}{\theta}$ sectors of boundary of communication range (Poduri *et al.*, 2009).

In the proposed deployment algorithm (Poduri *et al.*, 2009), repelling and attracting forces between mobile nodes are used. These forces have inverse square law profiles. Repelling force tends to be infinity when distance between nodes reaches zero, while attracting force tends to be infinity when the distance increases to R_c (an upper bounder of edge length). Since all mobile nodes are assumed to be initially densely deployed, the network is well connected and the NET condition can be satisfied. Each node repels its neighbors to increase the sensing coverage. During this process, some neighbors become unreachable if they move farther than CR to the node. Once the number of neighbors is close to the desired number of NET condition, a node assigns priorities to each of their neighbors based on their contribution toward satisfying the NET condition. A node assigns higher priority to the neighbor that contributes to a larger sector angle, and decides for each whether or not to apply repelling/attracting or both forces. Algorithm remains incomplete despite clear hints on its behavior.

Garetto *et al.* (2007) proposed a localized event-driven self-deployment algorithm. In this algorithm, a node receives virtual forces including exchange force from neighbors, potential force from detected events, and friction force subject to its velocity. All these forces are vectors, and together drive the node to move. A node k exerts exchange force on another node i if and only if k is neighboring i, $|ki| \neq 2r_s$, and there is no other node k' making $|k'i| < |ki| \wedge \angle kik' < \pi/6$. This condition limits the number of neighbors acting on node i to maximally six and forces the final network to have a TT layout. Exchange force is repulsive if nodal separation is less than $2r_s$, or attractive otherwise. Potential force can also be attractive or repulsive depending on a node's detected event intensity. This force pulls distant nodes toward the event location and pushes nearby nodes away. By adjusting event intensity threshold, different node density can be achieved around the event location. Friction force, which is always against nodal moving direction, is used to stop nodal movement so that a static equilibrium status can be eventually reached.

The strength of vector-based sensor self-deployment approach is that it enables nodes to make their deployment decision using solely their local knowledge. Some add-on techniques, for example, Voronoi-based termination technique (Wang *et al.*, 2004a) that requires global computation, may, however, offset this strength. This approach has many weaknesses in nature. Sensors cannot pass through closely placed obstacles due to their generated repulsive vector, resulting in sensing holes and coverage waste. Because node disappearance may break the equilibrium and trigger a chain of node movement (possibly network-wide) to recover, frequent topology change (possibly network-wide) may occur when node failure is a common phenomenon.

10.6.2 Voronoi-Based Approach

Use of Voronoi diagram for sensor self-deployment has been considered in the literature (Heo and Varshney, 2005; Wang *et al.*, 2004a; Cortes *et al.*, 2004). Voronoi diagram (Aurenhammer and Klein, XXXX) is a computational geometry structure widely employed in different fields. It partitions a plane using n given nodes $p_1, \ldots, p_n$ into n Voronoi regions, each containing exactly one node as generating node. The Voronoi region, V_i, of node p_i is the region of points that are closer to p_i than to any other. Namely, $V_i = \{q \in Q| \; ||q - p_i|| \leq ||q - p_j||, \forall j \neq i\}$, where Q represents the entire plane. Three Voronoi diagrams are shown in Figure 10.6.

The idea of Voronoi-based self-deployment is simple: sensors move to minimize their local uncovered areas (equivalently speaking, to maximize their sensing-effective areas) by aligning their sensing range with their Voronoi regions as much as possible. Usually, this approach involves multiple rounds of alignment and terminates when no more gain (e.g., utility gain in Heo and Varshney (2005) and coverage gain in Wang *et al.* (2004a)) can be achieved. Existing Voronoi-based algorithms differ merely in their node alignment methods. In Heo and Varshney (2005), a node moves to the point that maximizes a utility metric, which is defined as the product of the node's effective area and the node's estimated lifetime. In Wang *et al.* (2004a), a node moves half of the communication range toward the furthest Voronoi vertex, or to a so-called minimax point. In Cortes *et al.* (2004), a node moves to the weighted centroid of its Voronoi polygon. Below, we elaborate on the work presented in Cortes *et al.* (2004).

Cortes *et al.* (2004) studied coverage control in robot networks. Coverage ability of a robot is defined by a function of its location and the desired utility. For simplicity, consider the case of covering a source, and maximizing total team coverage for the source. Area coverage remains important, as utility function represents robot network's ability to cover events near the source. Each robot is assumed to know its own and its Voronoi neighbors' locations. It is also responsible for measurements within its Voronoi region. The goal is to control movement of robots to maximize detection probability. For example, detection probability of an event may decrease with squared distance to the event.

Robots move from their initial to final positions that optimize their collective monitoring ability. The proposed algorithm (Cortes *et al.*, 2004) runs in iterative fashion. While moving, robots update their Voronoi polygons. Centroids of Voronoi polygons are computed based on the Gaussian density function, representing reduced monitoring ability for a position further from the source. Therefore, these weighted centroids in each Voronoi polygon tend to be closer to

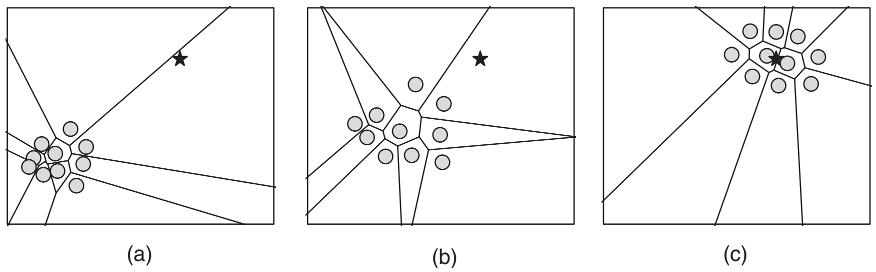


Figure 10.6 Voronoi-based approach: (a) initial distribution; (b) transient distribution; and (c) final distribution.

the source than the geometric center of Voronoi polygon (which is at the current robot position). Their locations also depend on the position of neighboring robots. Robots move toward the centroids of their corresponding Voronoi polygons. They are expected to converge toward the final position. In the example in Figure 10.6, the source is marked by black star. Initial positions and final positions of robots are shown in Figure 10.6a and 10.6c, respectively; a transient node distribution is given in 10.6b.

In Voronoi-based approach, Voronoi diagram needs to be repeatedly constructed to reflect nodal movement. Since Voronoi diagram construction requires global computation, this approach has large message overhead. To avoid oscillations (i.e., moving back and forth between several points), in these algorithms, nodes may stop their movement early according to certain policies. Early stop may, however, bring coverage redundancy and hole presence.

10.6.3 Load Balancing Approach

Yang *et al.* (2007) presented a distributed load balancing algorithm for sensor self-deployment. This algorithm partitions the target field into a 2D mesh, and treats nodes as load. The objective is to balance the load, that is, the number of nodes, in each mesh cell. Figure 10.7a shows load (also called *weight*) of each cell of a 6 × 6 mesh partition. By this algorithm, nodes in each mesh cell form a cluster covering the cell and are managed by an elected clusterhead.

During a preprocessing phase, a recursive doubling expansion is performed first along mesh columns and then along mesh rows. The objective of this phase is to fill empty clusters. Expansion on different columns (or rows) can proceed in parallel. Specifically, the maximum sequence of nonempty clusters in a column or a row expands toward one direction by planting a seed (i.e., a node) in its neighboring empty clusters in iterations. The initial span of expansion is subject to the weight (i.e., number of nodes) of the sequence; in each iteration, it doubles the span of its previous expansion. This expansion stops when the last cluster is covered or when there are no spare nodes left. An expansion toward the opposite direction may start afterwards, if applicable.

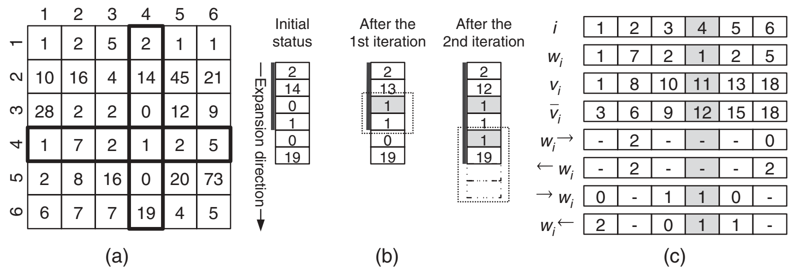


Figure 10.7 Load balancing approach: (a) node distribution; (b) doubling expansion; and (c) scan.

Observe Figure 10.7b that shows the doubling expansion of column 4 of the mesh in Figure 10.7a. The first two clusters (cells), marked by thick line in Figure 10.7a, constitute the initial maximum sequence of nonempty clusters. This sequence would to expand to fill the empty clusters below it. The doubling expansion reaches column end and terminates after two iterations; it is shown by the growth of the thick line. In the first iteration, the sequence grows by two clusters. In the second iteration, it attempts to grow by four clusters, but the expansion stops halfway because the end is reached. In the two iterations, a seed is planted in the third cluster and the fifth cluster, respectively. Afterwards, the doubling expansion terminates, and no empty cluster exists.

After the preprocess phase, a scan phase starts for load balancing. This phase is executed in two rounds. Every mesh row is scanned and load balanced in the first round; every mesh column is processed during the second round. In a scan round, load balancing along different rows (or columns) can proceed in parallel. For simplicity, let us consider a 1D array that maps to either a mesh row or a mesh column.

During a scan round, the algorithm first scans the array from one end to the other. In this scan, each cluster i (actually its clusterhead), whose weight is denoted by w_i, computes prefix weight sum $v_i = v_{i-1} + w_i$ of its previous clusters and passes v_i to the next cluster; the last cluster will compute the total weight of the array, and trigger another scan by sending the array weight back to the origin. In this scan, each cluster i computes the average cluster weight $\overline{w}$ and its prefix weight $\overline{v}_i = i\overline{w}$ in balanced status, and then determines its status (overloaded or underloaded) and the number of nodes (load) to send to/take from each direction along the array. If $w_i > \overline{w}$, it is overloaded, and the numbers of nodes it needs to give to the right and the left are, respectively, $w_i^{\rightarrow} = \min\{w_i - \overline{w}, \max\{v_i - \overline{v}_i, 0\}\}$ and ${}^{\leftarrow}w_i = (w_i - \overline{w}) - w_i^{\rightarrow}$. If $w_i < \overline{w}$, it is underloaded, and the numbers of nodes it needs to take from the left and the right are, respectively, ${}^{\rightarrow}w_i = \min\{\overline{w} - w_i, \max\{v_{i-1} - \overline{v}_{i-1}, 0\}\}$ and $w_i^{\leftarrow} = (\overline{w} - w_i) -{}^{\rightarrow} w_i$.

Figure 10.7c shows the results of the scan phase of row 4 of the mesh given in Figure 10.7a. Focus on the fourth cluster in this row, that is, the case of $i = 4$. After the first scan, it (in fact, its clusterhead) knows $v_3 = (1 + 7) + 2 = 10$ and thus its prefix weight sum $v_4 = 10 + 1 = 11$. In the second scan, when it receives the array weight 18 from right, it calculates $\overline{w} = 18/6 = 3$ and $\overline{v}_4 = 4 * 3 = 12$. By comparing its own weight w_4 and the average weight $\overline{w}$, it realizes that it itself is underloaded. Then it computes ${}^{\rightarrow}w_i = \min\{3 - 1, \max\{10 - 3 * 3, 0\}\} = \min\{2, \max\{1, 0\}\} = 1$ and $w_i^{\leftarrow} = (3 - 1) - 1 = 1$. From the results, it knows that it should take one node from each direction when nodes flow through it.

This approach requires the network to be dense enough so that load balancing can be proceeded in the entire sensory field. As the authors admitted, it may generate huge message overhead when the network is very dense due to the increased number of rounds of scans.

10.6.4 Stochastic Approach

Mousavi *et al.* (2006) proposed a localized stochastic deployment algorithm, stanastic deployment routine (SDR). In this algorithm, sensors are dropped in a $X \times Y$ field, and they move from dense area to sparse area according to their local knowledge through restricted random walk. The execution of SDR is independent of network connectivity. Because of its stochastic nature, SDR provides no guarantee on coverage maximization, hole elimination, or connectivity in the final network.

In SDR, time is divided into successive epochs of same size. A sensor node moves at local epoch t toward a location randomly and uniformly picked within a *moving rectangle* (MR_t). The position of MR_t changes as the node moves, and its size exponentially decreases as t increases. For $t \geq 0$, the east-to-west width of MR_t is $X \cdot p_0 \cdot p^t$, while its north-to-south width is $Y \cdot p_0 \cdot p^t$, where $0 < p_0 < 1$ and $0 < p < 1$ are predefined constants. Notice that the size of MR_t depends on time t only. Restricted by the ever-shrinking moving rectangle, the maximum moving distance of a sensor for a time unit exponentially decreases over time, which guarantees algorithm termination. The key is determination of the position of MR_t for each node at every epoch t. As we shall see below, this is accomplished locally by each node on the fly.

Suppose, at local epoch t, that a node has N neighbors (k-hop neighbors for a constant k) in total. Let N_w and N_e be the number of neighbors that are located, respectively, on the west and the east of the north-south line through the node; N_n and N_s are defined as the number of neighbors located, respectively, on the north and the south of the east-west line through the node. Then, $N = N_w + N_e = N_n + N_s$. For example, in Figure 10.8, when the solid node is at position a, $N = 19$, $N_w = 10$, $N_e = 9$, $N_n = 13$, and $N_s = 6$. Further, let d_w, d_n, d_e and d_s denote the distance from the node, respectively, to the west-, the north-, the east-, and the south border of its MR_t. As each node is always expected to move to sparse area from dense area, the MR_t should be positioned in such a way that it covers a small part of the local dense area of the node. Hence, it is defined that $\frac{d_w}{d_e} = \frac{N_e}{N_w}$, and, in this case, $\frac{d_w}{d_e+d_w} = \frac{N_e}{N_w+N_e}$, and therefore $d_w = \frac{N_e}{N} MR_{t \cdot x}$, where $MR_{t \cdot x}$ represents the width of the MR_t. Likewise, $d_n = \frac{N_s}{N} MR_{t \cdot y}$.

At every local epoch t, each node is able to compute the size of the MR_t and its relative position within MR_t, that is, d_w, d_n, d_e, and d_s, and therefore it is also able to compute the exact position of MR_t given its own geographic location. After MR_t is computed, the node picks a target location within its size-reduced MR_t probabilistically with uniform distribution and move all the way to that location. Then at local epoch $t + 1$, the node repeats the computation and movement with respect to its new k-hop neighborhood information. Figure 10.8 shows the movement steps and the MR_t of a node for $t = 0, 1, 2$. Note, if the node finds that the number of neighbors is equal to 0 (or less than a specific number), then it will cancel its movement (respectively, reduce the range of its movement). Two neighboring sensors may exchange their target location if they

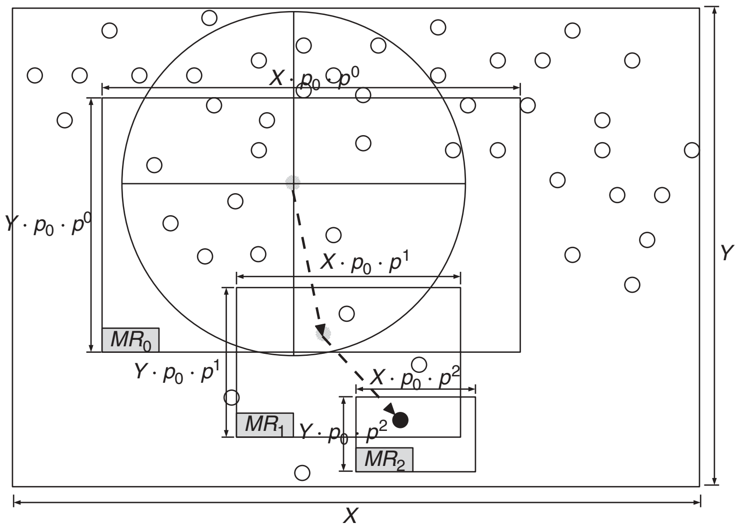


Figure 10.8 MR-based stochastic movement.

find that doing so will reduce their moving distance and thus energy consumption. Finally, every node stops moving when the size of its moving rectangle becomes too small, for example smaller than a threshold value.

10.6.5 Point-Coverage Approach

Tree-Based Point Assignment

Mousavi *et al.* (2006) presented a distributed one-step deployment (OSD) algorithm under the assumption of $r_c \geq \sqrt{2}r_s$. This algorithm partitions the ROI evenly into two-dimensional square grids, each with edge length $\sqrt{2}r_s$, and instructs sensors to occupy all the grid points. The intuition is that if every grid point is occupied by a sensor, then the entire ROI is fully covered, and meanwhile the sensors form a connected network.

In OSD, a breath-first tree rooted at an elected node is established first, and a converge process is then initiated by leaf nodes. In the converge process, after receiving a message containing the size of the corresponding subtree from all its children, a node computes the size of the subtree rooted at itself and sends the information to its parent. After the converge process, each node knows the size of each of its subtree. Thereafter, a recursive vertex assignment process starts. Specifically, the root chooses grid point (0,0) as its own deployment destination and assigns each of its subtrees a subarea with a matching number of grid points. The root of each subtree does the assignment in the same way. This recursive assignment stops when leaf nodes are reached. Finally, each node knows about its designated deployment point and moves there by one step to construct a full coverage. This algorithm saves energy by one-step movement strategy. However, it requires connectivity of all mobile sensors at the beginning, root election, and tree construction overhead.

Snap and Spread

Bartolini *et al.* (2008a) presented a snap and spread self-deployment algorithm, under the implicit assumptions that the ROI is bounded (boundary information is not known a priori though) and that there are sufficient sensors to cover the entire ROI. This algorithm arranges sensors at hexagon centers of a hexagonal grid, where hexagon edge length is equal to r_S. Spontaneously, mobile sensor starts to construct a hexagonal tiling over the ROI by choosing its current position as the center of the first hexagon of the tiling, changing its status to snapped and assigning itself order 0.

A snapped sensor learns the status of its neighbors through local communication and establishes a slave set containing unsnapped neighbors in its hexagon. It pushes its slaves to the center of adjacent empty hexagons. Those slaves then become snapped and are assigned an order larger than their master's by 1. This snap activity is illustrated in Figure 10.9a, where node 1 snaps its slaves, that is, nodes 2–9, to neighboring hexagon centers.

After the snap activity, if there are still spare sensors in its hexagon, the sensor starts a spread process, where it pushes these slaves to the adjacent hexagons with less sensors and a higher order, as shown in Figure 10.9b. If multiple such hexagons exit, closest one is selected. By this means, redundant sensors are always pushed to expand the boundary of the hexagonal tiling. A

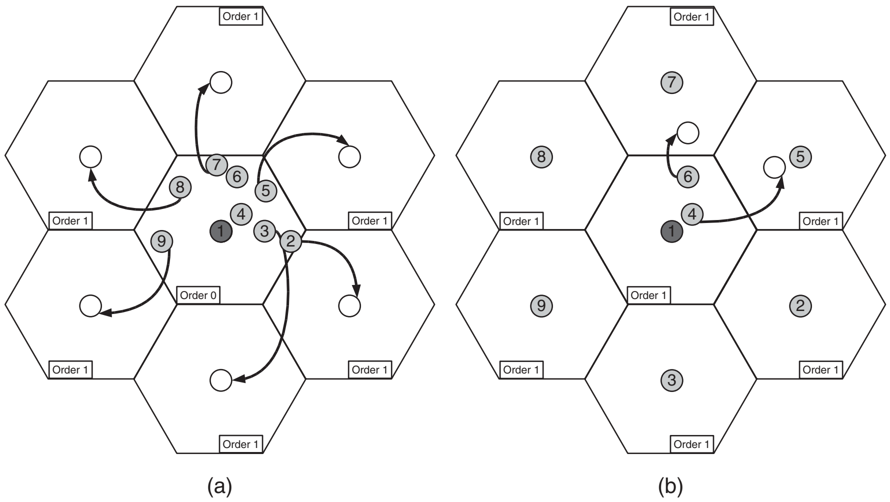


Figure 10.9 (a) Snap and (b) spread approach.

snapped sensor with adjacent empty hexagon(s) starts a pull process which is in essence expanding ring search and attract unsnapped sensors from remote hexagons.

As multiple sensors start the algorithm independently, multiple tiling portions may exist. These tiling portions may not align with each other because they start from arbitrary points. When two tiling portions meet, the one that started earlier absorbs the other. The position of snapped sensors in the absorbed tiling portion is adjusted. The adjustment starts at the frontier where the two portions meet and propagate to the entire tiling portion.

Compared with already snapped sensors, unsnapped ones consume relatively larger amount of energy on communication and movements for finding proper deployment positions. To balance energy consumption among sensors, they may exchange their role from time to time. Density control can be accomplished by forbidding snap and spread activities when the node density in target hexagon is lower than a density threshold.

The algorithm is not purely localized because, according to the implementation presented in Bartolini *et al.* (2008b), in a pull process for filling adjacent empty hexagons, a snapped sensor has to visit (by sending a message) every other hexagon in the worst case before finding an unsnapped sensor.

Combined Greedy-Rotation

Li *et al.* (2008a, 2009) introduced *focused coverage* problem. The sensor area coverage with a focus on covering a given point of interest (POI) is called *focused converge*. It is measured by *coverage radius*, that is, the minimum distance from the POI to uncovered areas. Optimal focused coverage has maximized coverage radius. The authors presented two purely localized sensor self-deployment algorithms abbtextgreedy advance (abbGAD) and greedy-rotation-greedy (GRG) for focused coverage formation.

Suppose that sensor nodes are randomly deployed in the coverage region and may possibly be disconnected at the beginning. Assume that $r_c \geq \sqrt{3}r_s$ and that sensors know their own geographic locations. The problem is to move sensor nodes to build a connected network surrounding the POI, denoted by P, with an equilateral TT layout (Fig. 10.10). The reason for employing TT layout is that this layout maximizes the coverage area of a given number of nodes without any sensing hole when nodes are placed on the vertices of the layout while guaranteeing connectivity of the network (Bai *et al.*, 2006; Ma and Yang, 2007; Zhang and Hou, 2005).

The basic idea of GA is to greedily move nodes along TT edges toward P. Each node located at a vertex of TT moves to another vertex which is closer (in graph distance) to P than its current location. There are at most two directions for movement of nodes based on the current location of nodes. The node located at a corner vertex has only one possible moving direction. Three rules were proposed for movement control. The first rule is to determine priority for simultaneous movement from two vertices to the same vertex. In Figure 10.10a, if two nodes are

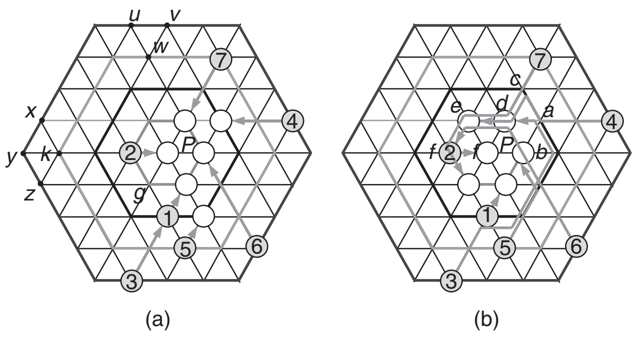


Figure 10.10 Equilateral triangle tessellation and focused coverage formation: (a) GA and (b) GRG.

moving to k respectively from x and y (or y and z), higher priority will be given to the node from y (respectively, z). However, to avoid potential simultaneous movement of nodes from x to z, the movement of a node from z to k is forbidden by the second rule. The third rule allows any node that is located on the hexagon adjacent to P to move to P as long as P is not occupied. Figure 10.10, where node trajectories are marked by arrow lines, illustrates how GA works. Note that, in this example, node 3 stops at the initial position of node 5. It does not move to vertex g even though g is empty because of the second rule.

GRG consists of the greedy advance movement and a new type of movement—rotation. Rotation is applied in a node when the node's greedy advance movement is blocked. It is to move the node to a predetermined direction, e.g. counterclockwise, at the same layer. Since the proposed algorithms do not require time synchronization at sensor nodes, rotation movement of nodes in a layer may block the greedy advance movement of nodes in the higher layer (further to P). Thus, a suspension rule is introduced to cancel rotation movement of a node when it detects there is a neighbor rotating at higher layer. In the case that a greedy advance movement and a rotation movement target at the same vertex, a competition rule is applied to give higher priority to greedy advance movement. Three more rules are proposed to cope up with movement of the nodes at special locations, such as corner nodes and gateway nodes.

The execution of GRG is illustrated in Figure 10.10b. Let us focus on nodes 2, 4, and 6. Node 2 moves to its final position, P, by a single greedy advance step, whereas nodes 4 and 6 travel a curly long path. Node 4, after reaching a by greedy advance, finds that d is occupied by node 7, and that its further greedy advance to b is forbidden. So it rotates around P along its residence hexagon. When node 4 rotates to c, node 6 arrives at b. At that moment, d becomes unoccupied as node 7 leaves, and so P would be taken by node 4. Then, node 2 decides to greedily proceed to d and node 6 decides to rotate to d, resulting in a collision at d. The rotating node 6 has higher priority to take the next deployment step and continues its rotation, while node 2 has to wait. Finally, node 6 rotates to its final position f, passing through e; node 2 rotates to e after node 6 leaves e for f.

It is proven that both GA and GRG terminate in finite time and yield a connected network with maximized hole-free coverage (Li *et al.*, 2009). In fact, they are the first localized sensor self-deployment algorithms that provide such guarantee. Simulation results show that GA has shorter convergence time and consumes less energy than GRG algorithm. In Li *et al.* (2009a), an optimized version of GRG, called *OGRG*, is presented by the same authors. OGRG uses deployment polygons best approximating circles rather than hexagons for rotation and thus yield guaranteed circular coverage radius maximization. In Li *et al.* (2009b), an improved version of GRG, referred to as GRG with Obstacle Penetration (*GRG/OP*), is presented. GRG/OP is equipped with a novel obstacle avoidance capability and shown to be able to solve not only focused coverage but also traditional area coverage.

10.6.6 Incremental Approach

Howard *et al.* (2002b) presented an incremental deployment algorithm for homogeneous mobile sensors with the ability to "see" their surroundings. The objective is to generate a connected network with maximized total area visible to the network while maintaining a line of sight among sensors. At initiation, all the nodes but one are considered to be undeployed, and the only deployed node serves as starting point. The algorithm runs on a central controller in iterations. In each iteration, the central controller deploys only one sensor to push the frontier line of the network forward toward unknown area. Below, let us examine an algorithm iteration.

In an algorithm iteration, the central controller gathers the information of previously deployed sensors and constructs an occupancy grid over the target field. In the occupancy grid, a cell is considered *free* if it contains no obstacle, or *occupied* if it contains obstacles, or *unknown* otherwise (i.e., if no knowledge whether this cell is available or if contradictory evidence about the cell's occupancy state exists). Figure 10.11a shows an occupancy grid constructed according to the information from the first node *a* deployed in an environment. In this figure, the only node's vision capability is marked by closed lines representing a circle; black cells are occupied, white cells are free, and the rests are unknown.

The central controller converts the occupancy grid to a configuration grid, where a cell is *free* if and only if all nearby (within certain predefined distance, e.g., cell size) cells are free, or *occupied* if at least one nearby cell is occupied, or *unknown* otherwise. Figure 10.11b shows the configuration grid corresponding to the occupancy grid in Figure 10.11a. In this figure, white cells are free cells, black cells are occupied cells, and the others are unknown cells.

The configuration grid is further transformed to a reachability grid. In this process, a free cell in the configuration grid is marked as *reachable* if there is some chain of free cells between this cell and the location of certain deployed node, or *unreachable* otherwise; any other cell is marked as *unreachable*. Figure 10.11c, where reachable cells are represented by white cells, shows the reachability grid obtained from the configuration grid in Figure 10.11b. Notice

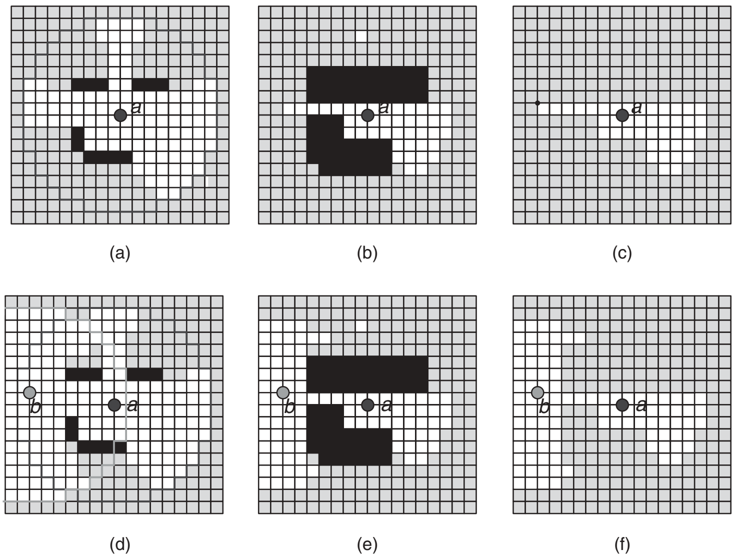


Figure 10.11 Incremental approach: (a) occupancy grid (1 node); (b) configuration grid (1 node); (c) reachability grid (1 node); (d) occupancy grid (2 nodes); (e) configuration grid (2 nodes); (f) reachability grid(2 nodes).

that the white cell above the black cells in Figure 10.11b are marked unreachable in Figure 10.11c because there is no free-cell path linking it to the node.

After the reachability grid is built, a node can be placed conservatively to a location between free and unknown space to minimize the overlapping of sensory field, or optimistically to a location where they can reduce maximally unknown space. In this process, candidate cells may not be unique. Different policies can be applied to guide the selection of reachable cells, yielding different network topologies at the end.

Once the target location is determined, a shortest path through previously deployed nodes between the entry point (the point from which nodes enter the environment) and the target location is discovered. A sequential or a concurrent shifted movement (Section 10.3) process is performed along the path. This method solves sensors' interblocking during their movement and balances energy consumption. Look at Figure 10.11c. Suppose the target location is the dotted point. Then, node *a* will move to that point while a new node *b* is dropped at *a*'s position (the entry point). Figure 10.11d–f show the occupancy grid, the configuration grid, and the reachability grid after node *b* is added into the environment.

10.6.7 Maximum-Flow Approach

Chellappan *et al.* (2007) presented a centralized minimum-cost, maximum-flow based motion planning algorithm for *hopping* sensors randomly dropped in a rectangular field. Sensors are able to move at most once up to distance $F = k * d$ for some constant k, where d is basic distance unit. The target field is divided evenly into an $R \times R$ grid, where R is a predefined region size. The goal is to maximize the number of covered regions using a minimal number of sensor moves. Consider regions with at least one sensor as *source* and empty regions as *holes*. The algorithm builds a virtual directed graph that records for each region the information of its hosted sensors and parameterizes interregion paths based on the desired objective and mobility constraints; it aims to maximize the flow of sensors from source regions to hole regions with minimized cost and without violating the path constraints in between. As there exist many maximum-flow and minimum-cost problem statements and solution approaches, it focuses on virtual directed graph construction. Graph construction is discussed in different cases.

10.6.8 Genetic-Algorithm Approach

Ramadan *et al.* (2007) modeled sensor placement as combinatorial optimization problem. They considered a set S of sensors for a target field composed of a number of zones A for a time horizon T. For every time interval $t \in T$, each sensor $s \in S$ is associated with a predefined time-evolving reliability R_s^t, and each zone $i \in A$ is assigned a time-varying weight function w_t^i defining the importance of the observations in the zone over T. The objective is to maximize coverage, that is, to ensure all the zones with highest weight to be monitored, and sensors with high reliability to be assigned to high weight zones. Alternatively, coverage is also considered maximized when the number of monitored zones is maximized, and each zone is monitored by exactly one sensor at any time.

The authors presented a heuristic motion planning algorithm based on genetic algorithm (GA). A GA algorithm simulates genes and nature selection. In general, it generates, usually at random, a preliminary set of chromosomes at an initialization step, and then executes the following steps in iterations until certain stopping criteria are met: (1) Selection: evaluate the fitness of each individual chromosome in current chromosome set and select the best ranked ones. (2) Reproduction: generate a new chromosome set by applying crossover and/or mutation operations on selected chromosomes to generate offsprings and using the offsprings to replace worst ranked chromosomes in current chromosome set.

A chromosome contains $|A| \cdot |T| \cdot |S|$ genes, each of which is assigned a truth value implying a deployment plan for a sensor s at certain time interval t in certain zone i. The initial chromosome set has predefined size n. It is generated either randomly or by certain given rules. Two crossover operations, time exchange (TE) and best chromosome (BC), are defined. They both exchange the sensor deployment patterns in the same time interval in two different chromosomes. TE uses two randomly selected chromosome; BC uses two fittest ones. The fitness of each chromosome x is measured by function: $F(x) = \sum_t \sum_i \sum_s w_i^t R_s^t$. Mutation operations, where some chromosome genes are randomly flipped, are performed after crossover operations to prevent searching dead end and chromosome repetition. A new chromosome is accepted iff it passes a feasibility check respecting the capability constraints of each sensor. The algorithm stops after a fixed number of iterations or if improvement does not seem possible.

A simple single-chromosome-per-iteration mechanism can be alternatively used. In this case, a chromosome set contains only one element, and TE operation is for exchanging the sensor deployment pattern in two randomly selected time intervals in the only chromosome. Since BC is no longer applicable, a new crossover operation sensor exchange (SE) is employed instead. By SE, the deployment pattern of two randomly selected sensors over the entire horizon T is exchanged.

10.7 SENSOR RELOCATION

To minimize energy consumption and response time, a replacement node should be a redundant sensor geographically closest to the failed node. Thus, replacement discovery is a distance-sensitive service discovery problem, where redundant sensors as service provider offer replacement service to failed sensors. After replacement discovery, discovered replacement will be migrated to the position of failed sensor. Replacement migration can be accomplished in a *direct* way or in a *shifted* manner (Section 10.3).

Many service discovery algorithms (Gao *et al.*, 2006; Mian *et al.*, 2006) have been proposed for wireless ad hoc networks. They can certainly be used to fulfill the replacement discovery problem. Some techniques such as location service (Chapter 8) and data centric storage (Li *et al.*, 2008c) can also be adopted. By location service, redundant sensors update the network with their location and are searched when needed. By data centric storage, the location data of redundant sensors are stored somewhere in the network and retrieved by others. But, considering the resource constraints of sensors, a good solution should have low message overhead and constant per node storage load.

There exist a few sensor relocation algorithms in the literature. In the following sections, we group these algorithms according to their employed replacement discovery methods and review them in detail.

10.7.1 Broadcast-Based Approach

Wang *et al.* (2004b) proposed a proxy-based sensor relocation protocol for networks composed of both static sensors and mobile sensors. Every mobile sensor periodically advertises itself by broadcasting within a predefined radius its current location and its base price, which, initially set to zero, reflects how much coverage contribution it is currently making. Static sensors construct a Voronoi diagram and listen to mobile sensors' advertisements. After receiving an advertisement, a static node records the embedded information and maintains a mobile sensor list. Once a static sensor detects a sensing hole in its Voronoi polygon, it estimates the hole size and computes a bid accordingly; then, it chooses from its mobile sensor list a closest one with lowest base price, which is smaller than the bid, and sends a bidding message to that sensor. In the case that a mobile sensor receives more than one bidding message from different static sensors, it chooses the highest bid and sends a delegate message to the corresponding bidder. After receiving the delegate message, the bidder becomes the proxy of the mobile sensor and executes the relocation protocol on its behalf as if the mobile sensor had migrated to the sensing hole.

Mobile sensors having proxies cease to execute the protocol and wait for a movement notification from their proxies. As the protocol executes, the proxy of a mobile sensor may change from one for a small sensing hole to one for a large sensing hole, rendering the mobile sensor logically migrating from one location to another. When a proxy node fails in finding larger sensing holes with respect to the base price of its delegated sensor, it will consider that the current logical location is the final location, and informs the delegated sensor to physically move to that location. To reduce moving distance, proxy nodes may exchange their delegated sensors. When a proxy node finds that its delegated sensor has to move a distance longer than a predefined threshold value, it searches its mobile sensor list for such a proxy node that the moving distance of their delegated sensors will be shorter than the threshold value if they exchange their delegated sensors. Then, it will send an exchange message to that node (if any) and wait for a confirmation message. In the exchange message, the moving distance without delegated sensor exchange is specified so that the receiver is able to make its decision on the proposal.

When different static sensors detect the same sensing hole, they bid mobile sensors independently and possibly cause multiple mobile sensors moving to the same location. To avoid such collision, a proxy node tries to re-detect the sensing hole that its delegated sensor is going to heal. If the hole still exists, it will simply consider that there is no collision. Otherwise, the proxy node will further check if the moving distance of its delegated sensor is the shortest among those of other mobile sensors for the same hole. If so, it waits for others to give up; otherwise, it cancels the movement, sets the base price of its delegated sensor to zero, and re-advertises the new price in the next round.

Figure 10.12, where square nodes and round nodes, respectively, represent static sensors and mobile sensors, illustrates how the algorithm works. The Voronoi diagram created using static nodes is also drawn in the figure. Nodes 6, 7, and 8 are located within the advertisement range (indicated by dashed circle) of node 1. Node 8 bids node 1 for its local sensing hole at position A, competing with nodes 6 and 7. It wins the competition, becomes the proxy of 1, and advertises on its behalf. Node 9 receives the advertisement from node 8 and then successfully bids node 1 for its local sensing hole at position B. After that, it becomes the new proxy of node 1. Similarly, node 10 takes over node 1 from node 9 for its local sensing hole at C. Then it finds that C is the final location for node 1 and informs it to move. After receiving the notification from node 10, node 1 moves directly to C in one step.

This protocol consumes a large amount of bandwidth for periodic broadcast-based advertising. It does not guarantee sensing hole filling, as it is always

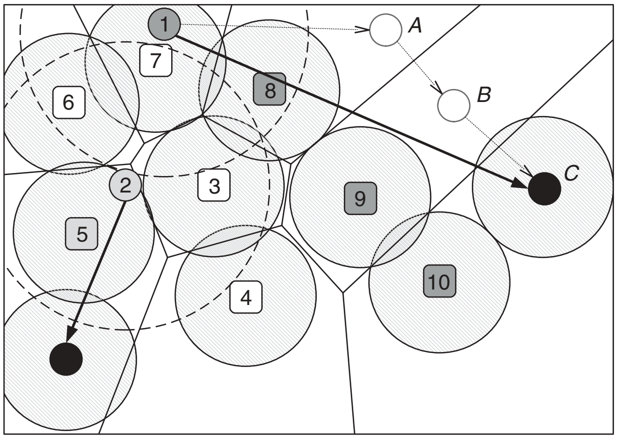


Figure 10.12 Broadcast-based approach.

possible that some static sensor with local sensing holes does not discover mobile sensors, unless the advertisement range covers the entire network.

10.7.2 Quorum-Based Approach

Wang *et al.* (2005) presented a grid-quorum-based relocation protocol. In this protocol, the sensory field is evenly partitioned into grids. In each grid, one node is elected as grid head and takes the responsibility to collect the information of all the grid members; based on grid members' location, a grid head determines redundant sensors and detects sensing holes. The grid heads in a row form a supply quorum. The grid heads in a column form a demand quorum. A grid head publishes the information about the redundant nodes inside its grid along its residing supply quorum. When any grid needs more sensors for hole healing, the grid head searches along its residing demand quorum to discover the closest redundant node. To reduce message complexity, information about already discovered closest redundant node is piggybacked on the search message and used to restrict the distance that the message may travel further. This protocol uses shifted replacement migration method. Migration path is constructed by a flooding process confined within an elliptic zone covering both the sensing hole and the redundant node.

The protocol does not address how to ensure replacement discovery in the presence of void areas (e.g., empty grids appearing in demand quorum and blocking search messages). It is hard to predetermine the size of the elliptic search zone so as to guarantee success of migration path discovery. On the other hand, setting the zone to the entire network can provide the guarantee, but induces increased message overhead.

Li and Santoro (2006) presented a zone-based sensor relocation protocol (ZONER) on the basis of the quorum technique. In ZONER, each predetermined redundant node publishes its location within a vertical registration zone across the network. After a nonredundant node fails, its west-most and east-most neighbor take as reference the closest redundant nodes they know, and request in bounded horizontal request zones, respectively, toward the west and the east, for a yet closer redundant node. In this request process, the nodes in the intersection area of a request zone and a registration zone can reply as recommender with requested information. Then the two requesters exchange their discovery results through underlying routing protocol to determine the replacement. Both node registration and node request are performed by a zone flooding technique, which is a combination of simple range-restricted flooding and face routing, and featured with the void-area-penetration ability. ZONER uses shifted migration method to relocate the replacement to the failure node's position along a natural migration path with an extra discovery process. The migration path is the aggregation of the registration path of the replacement to the recommender and the request path of the discoverer to the recommender.

Although ZONER (Li and Santoro, 2006) and the protocol presented in Wang *et al.* (2005) have similarity in their node discovery and node migration methods, they differ in that ZONER requires no preknowledge of the sensor field and guarantees replacement discovery (by resorting to face routing) and node replacement.

10.7.3 Mesh-Based Approach

Li *et al.* (2007) proposed a mesh-based sensor relocation protocol (MSRP) on the basis of a localized distance-sensitive service discovery protocol iMesh (Li *et al.*, 2009c) (described in Chapter 8). In MSRP, each redundant node (R-node) spontaneously takes a nearest active node (A-node) as proxy, by sending it a delegation request. Proxy nodes execute iMesh to construct an information mesh. While being awake, an R-node monitors the aliveness of its proxy; once it finds that its proxy fails, it moves to replace the proxy node directly. Upon an ordinary (i.e., nonproxy) A-node failure, the A-nodes neighboring the failed A-node cooperate to discover a replacement, which is defined as the nearest delegated R-node of the target proxy (referred to as *replacement proxy*) of the failed A-node, by iMesh. During a replacement discovery process, the north-most, the south-most, the east-most and the west-most neighbor of a failed A-node, as *server*, send a query message, respectively, to the north-, the south-, the east-, and the west-direction. After getting replies, they exchange their discovery results through underlaying routing protocol to find the replacement proxy.

The server closest to the replacement proxy is considered *replacement discoverer*. It issues a migration request to the replacement proxy, which then grants the request by an ACK message. After receiving the ACK, the replacement discoverer (or discoverer for short) starts a shifted migration process by sending an action message to the replacement proxy. During this process, the action message is transmitted by GFG (Bose *et al.*, 1999), coupling with the concept of cost over progress ratio (Stojmenovic, 2006), to establish a migration path from the discoverer to the replacement proxy, and, meanwhile, the intermediate nodes along this path shift their position toward the failed A-node. More specifically, after sending the action message, the discoverer moves to the failure node's position, while intermediate nodes move to the position of their prior hop after forwarding the action message. After receiving the action message, the replacement proxy first informs the replacement node to fill its current position and then itself moves toward the location of its prior hop.

10.7.4 Hierarchical-Structure-Based Approach

Jiang and Wu (2008) presented a hierarchical Hamilton cycle–based sensor relocation protocol. It is actually a variant of the hierarchical home-region-based location service (described in Chapter 8). In this algorithm, sensors have the same communication radius r_c and sensing radius r_s ($r_s = r_c$), and the sensory field is partitioned evenly into a number of $r \times r$ grids where $r = \frac{1}{\sqrt{5}} r_c$ (this r enables each node to communicate with nodes in neighboring grids directly). By this partition, both network connectivity and full coverage are preserved as long as there exists one node in each grid.

In a unit grid, one node is elected as grid head, while the others are grid members and considered redundant. Four neighboring unit grids form a level-1-directed Hamilton cycle in counterclockwise direction, and one of them is elected as the "eye" of the Hamilton cycle. The head of the eye grid collects the information on the existence of redundant nodes in the four unit grids along the cycle. The eyes of four neighboring level-1 Hamilton cycle further form a level-2-directed Hamilton cycle and share information. This Hamilton cycle formation is performed recursively until a level-k cycle covering the entire sensory field is built on four level-($k-1$) eyes. Any change of redundant nodes in a unit grid is monitored by the head of the unit grid, and will be collected by the grid head and then updated upward in the hierarchical Hamilton cycle structure.

When a grid head u finds that the neighboring grid in the direction of its residing level-1-directed Hamilton cycle is empty, it starts an intralevel repair process to fix the detected vacancy. It selects a redundant node in its grid to move to the empty grid. If no local redundant node is available, u itself will move to the empty grid; before moving, it sends a notification to the head v of its preceding grid in its residing level-1 Hamilton cycle. Upon notification, v repeats the above process for the grid of u, leading to shifted node migration. If the head w of the eye of the residing level-1 Hamilton cycle of u finds the lack of redundant nodes for a requested local hole repair in its dominated area (after receiving the notification), it will start an interlevel repair process.

In the interlevel repair process, w searches for a redundant node along its residing level-2 Hamilton cycle, and then continues until a redundant node is found at a level-i eye ($i \leq k$). Then, a localization process for the redundant node would start at level-i eye. It will proceed along the corresponding level-i cycle and reach a level-($i-1$) eye that has at least one redundant node in its dominated area. This process continues going down until reaching a unit grid. Then, from that unit grid, a redundant node is migrated in shifted way, along the path that is constructed in the above interlevel search and downward localization process, to fill in the detected empty grid.

REFERENCES

Akyildiz I, Su W, Sankarasubramaniam Y, Cayirci E. "Wireless sensor networks: a survey". Comput Netw 2002;38(4):393–422.

Aurenhammer F, Klein R. "Voronoi diagrams"; 2009. Available at http://www.pi6.fernuni-hagen.de/publ/tr198.pdf.

Bai X, Kumary S, Xuan D, Yun Z, Lai TH. "Deploying wireless sensors to achieve both coverage and connectivity". Proceedings of the 7th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc); Florence, Italy; 2006. pp. 131–142.

Bartolini N, Calamoneri T, Fusco EG, Massini A, Silvestri S. "Snap and spread: a self-deployment algorithm for mobile sensor networks". Proceedings of the 4th IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS); Santorini Island, Greece; 2008a. pp. 451–456.

Bartolini N, Massini A, Silvestri S. "P&P protocol: local coordination of mobile sensors for self-deployment"; 2008b. Available at http://arxiv.org/PS_c ache/arxiv/pdf/0805/0805.1981v1.pdf.

Batalin MA, Sukhatme GS. "The analysis of an efficient algorithm for robot coverage and exploration based on sensor network deployment". Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA); Barcelona, Spain; 2005. pp. 3478–3485.

Bose P, Morin P, Stojmenovic I, Urrutia J. "Routing with guaranteed delivery in Ad hoc wireless networks". Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M 1999); Seatle, Washington, USA; 1999. pp. 48–55.

Burgard W, Moors M, Fox D, Simmons R, Thrun S. "Collaborative multi-robot exploration". Proceedings of IEEE International Conference on Robotics and Automation (ICRA); San Francisco, CA; 2000. pp. 476–481.

Chang C-Y, Chang H-R, Hsieh C-C, Chang C-T. "OFRD: obstacle-free robot deployment algorithms for wireless sensor networks". Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC); Hong Kong, China; 2007. pp. 4371–4376.

Chellappan S, Bai X, Ma B, Xuan D. "Mobility limited flip-based sensor networks deployment". IEEE Trans Parallel Distrib Syst 2007;18(2):199–211.

Cortes J, Martinez S, Karatas T, Bullo F. "Coverage control for mobile sensing networks". IEEE Trans Rob Autom 2004;20(2):243–255.

Fletcher G, Li X, Nayak A, Stojmenovic I. "Back-tracking based sensor deployment by a robot team". A manuscript, 2009.

Fredslund J, Mataric MJ. "Robot formations using only local sensing and control". Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA); Banff, Alberta, Canada; 2001. pp. 308–313.

Gao Z, Yang Y, Zhao J, Cui J, Li X. "Service discovery protocols for MANETs: a survey". Proceedings of the 2nd International Conference on Mobile Ad-hoc and Sensor Networks (MSN) (LNCS 4325); Hong Kong, China; 2006. pp. 232–243.

Garetto M, Gribaudo M, Chiasserini C-F, Leonardi E. "A distributed sensor relocation scheme for environmental control". Proceedings of the 4th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS); Pisa, Italy; 2007.

Heo N, Varshney PK. “Energy-efficient deployment of intelligent mobile sensor networks”. IEEE Trans Syst Man Cybern A Syst Hum 2005;35(1):78–92.

Howard A, Mataric MJ, Sukhatme GS. “Mobile sensor network deployment using potential fields: a distributed, scalable solution to the area coverage problem”. Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS); Fukuoka, Japan; 2002a. pp. 299–308.

Howard A, Mataric MJ, Sukhatme GS. “An incremental self-deployment algorithm for mobile sensor networks”. Auton Robots 2002b;13(2):113–126.

Jiang Z, Wu J. “A hierarchical structure based coverage repair in wireless sensor networks”. Proceedings of the 19th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC); Cannes, France; 2008.

Li X, Frey H, Santoro N, Stojmenovic I. “Localized sensor self-deployment with coverage guarantee”. ACM SIGMOBILE Mobile Comput Commun Rev 2008a;12(2):50–52.

Li X, Frey H, Santoro N, Stojmenovic I. “Focused coverage by mobile sensor networks”. Proceedings of the 6th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS); Macau, China; 2009. To appear.

Li X, Lu K, Santoro N, Simplot-Ryl I, Stojmenovic I. “Alternative data gathering schemes for wireless sensor networks”. Proceedings of International Conference on Relations, Orders and Graphs: Interaction with Computer Science (ROGICS); Mahdia, Tunisia; 2008c. pp. 577–586.

Li X, Frey H, Santoro N, Stojmenovic I. “Localized sensor self-deployment for guaranteed circular coverage radius maximization”. Proceedings of the IEEE International Conference on Communications (ICC); 2009a. To appear.

Li X, Mitton N, Ryl I, Simplot D. “Localized sensor self-deployment with coverage guarantee in complex environment”. Proceedings of the 8th International Conference on AD-HOC Networks & Wireless (AdHoc-Now) (LNCS 5793) Murcia, Spain; 2009b. pp. 138–151.

Li X, Santoro N, Stojmenovic I. “Localized distance-sensitive service discovery in wireless sensor and actor networks”. IEEE Trans Comput 2009c;58(9):1275–1288.

Li X, Santoro N. “ZONER: a ZONE-based sensor relocation protocol for mobile sensor networks”. Proceedings of the 6th IEEE International Workshop on Wireless Local Networks (WLN); Tempa, USA; 2006. pp. 923–930.

Li X, Santoro N, Stojmenovic I. “Mesh-based sensor relocation for coverage maintenance in mobile sensor networks”. Proceedings of the 4th International Conference on Ubiquitous Intelligence and Computing (UIC); Hong Kong, China; 2007. pp. 696–708.

Lopez-Sanchez M, Esteva F, De Mantaras RL, Sierra C, Amat J. “Map generation by cooperative low-cost robots in structured unknown environments”. Auton Robots 1998;5(1):53–61.

Ma M, Yang Y. “Adaptive triangular deployment algorithm for unattended mobile sensor networks”. IEEE Trans Comput 2007;56(7):946–958.

Martincic F, Schwiebert L. “Introduction to wireless sensor networking”. In: Stojmenovic I, editor. Handbook of sensor networks. Hoboken (NJ) Wiley; 2005. Chapter 1, pp. 1–40.

Mei Y, Xian C, Das S, Charlie Hu Y, Lu Y-H. “Sensor replacement using mobile robots”. Comput Commun 2007;30(13):2615–2626.

Mian AN, Beraldi R, Baldoni R. “Survey of service discovery protocols in mobile Ad hoc networks”. Technical Report 4/06. Rome, Italy: Universit degli Studi di Roma La Sapienza; 2006.

Mitton N, Simplot-Ryl D, Stojmenovic I. “Guaranteed delivery for geographical anycasting in wireless multi-sink sensor and sensor-actor networks”. Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM); Rio de Janeiro, Brazil; 2009. To appear.

Mousavi H, Nayyeri A, Yazdani N, Lucas C. “Energy conserving movement-assisted deployment of Ad hoc sensor networks”. IEEE Commun Lett 2006;10(4):269–271.

Poduri S, Pattern S, Krishnamachari B, Sukhatme GS. “Using local geometry for tunable topology control in sensor networks”. IEEE Trans Mobile Comput 2009;8(2):218–230.

Ramadan R, El-Rewini H, Abdelghany K. “Optimal and approximate approaches for deployment of heterogeneous sensing devices”. EURASIP J Wirel Commun Netw 2007;2007(1):36.

Ratnasamy S, Karp B, Yin L, Yu F, Estrin D, Govindan R, Shenker S. “GHT: a geographic hash table for data-centric storage in sensornets”. Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA); Atlanta, Georgia, USA; 2002. pp. 78–87.

Scheider FE, Wildermuth D, Wolf H-L. “Motion coordination in formations of multiple mobile robots using a potential field approach”. In: Parker LE, Bekey GW, Barhen J, editors. Volume 4, Distributed autonomous robotics systems. Secaucus (NJ): Springer; 2000. pp. 305–314.

Stojmenovic I. “Localized network layer protocols in wireless sensor based on optimizing cost over progress ratio”. IEEE Netw 2006;20(1):21–27.

Sugihara K, Suzuki I. “Distributed algorithms for formation of geometric patterns with many mobile robots”. J Rob Syst 1996;13(3):127–139.

Wang G, Cao G, Porta TL. “Movement-assisted sensor deployment”. Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM); Hong Kong, China; Volume 4; 2004a. pp. 2469–2479.

Wang G, Cao G, La Porta T. “Proxy-based sensor deployment for mobile sensor networks”. Proceedings of the 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS); Fort Lauderdale, Florida, USA; 2004b. pp. 493–502.

Wang G, Cao G, La Porta T, Zhang W. “Sensor relocation in mobile sensor networks”. Proceedings of the 24rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM); Miami, Florida, USA; 2005. pp. 2302–2312.

Yamaguchi B, Shultz A, Adams W. “Mobile robot exploration and map-building with continuous localization”. Proceedings of IEEE/RSJ International Conference on Robotics and Automation (ICRA); Leuven, Belgium; 1998. pp. 3715–3720.

Yang S, Li M, Wu J. “Scan-based movement-assisted sensor deployment methods in wireless sensor networks”. IEEE Trans Parallel Distrib Syst 2007;18(8):1108–1121.

Zhang H, Hou JC. “Maintaining sensing coverage and connectivity in large sensor networks”. Ad Hoc & Sens Wirel Netw 2005; 1(1–2):89–124.

Index