4

ROUTING AND DATA DISSEMINATION

Sajal K. Das

University of Texas at Arlington, USA

Habib M. Ammari

Hofstra University, USA

4.1 INTRODUCTION

Routing and data dissemination are an important issue in wireless sensor networks (WSNs). The essential function of a WSN is to monitor a phenomenon in a physical environment and report sensed data to a central node called a *sink*, where additional operations can be applied to the gathered data. This chapter focuses on routing and data dissemination in WSNs, and introduces the fundamental concepts related to routing and data dissemination, discusses the major issues and challenges in accomplishing this vital function, and surveys a variety of protocols for routing and data dissemination in WSNs. In particular, we present a taxonomy of routing and data dissemination protocols for WSNs based on different classification criteria, for example, location information, network layering and in-network processing, data centricity, multipath, network dynamics, quality-of-service requirements, and heterogeneity. The taxonomy is developed through an extensive analysis of a variety of routing and data dissemination protocols for WSNs. The objective of the taxonomy is threefold: (1) to provide a framework

Wireless Sensor Networks: A Networking Perspective, Edited by Jun Zheng and Abbas Jamalipour

in which routing and data dissemination protocols for WSNs can be examined and compared; (2) to show how the routing and data dissemination protocols can be categorized according to this taxonomy; and (3) to gain new insights into the routing and data dissemination protocols and thereby suggest avenues for future research. More specifically, the taxonomy comprises two types of classifications: one that classifies routing and data dissemination protocols with respect to sensor deployment, for example, sensor mobility, where sensors could be mobile or static, and one that classifies them with respect to trade-offs between different metrics specific to sensing applications, for example, energy efficiency, low delay, high data accuracy, and fault tolerance. Also shown are the benefits of sensor heterogeneity in routing and data dissemination for WSNs. This chapter complements other existing excellent surveys on WSNs [1,2], as well as those on routing and data dissemination protocols for WSNs [3–5].

The remainder of this chapter is organized as follows: Section 4.2 introduces the fundamentals and presents the major challenges in routing and data dissemination in WSNs. Section 4.3 overviews the ingredients of interest of the taxonomy for a variety of existing protocols. Section 4.4 surveys a sample of existing routing and data dissemination protocols in WSNs and classifies them with respect to the taxonomy. Section 4.5 concludes this chapter.

4.2 FUNDAMENTALS AND CHALLENGES

This section introduces related fundamentals and presents the major challenges in the design of routing and data dissemination protocols for WSNs.

4.2.1 Fundamentals

First, we define the terminologies that will be used in the subsequent sections. Then, we introduce a commonly used energy model [6] in most of the protocols for WSNs. Finally, we describe the *Voronoi* diagram [7], which has been widely used as a model of WSNs.

4.2.1.1 Terminology.

Sensing Range. The *sensing range* of a sensor (s_i) is a disk of radius (r_i), including its boundary, centered at ξ_i (the location of s_i) and defined by the point set, $D(\xi_i, r_i) = \{\xi \in \mathbf{IR}^2 : |\xi_i - \xi| \leq r\}$, where $|\xi_i - \xi|$ is the Euclidean distance between the locations ξ_i and ξ.

Transmission Range. The *transmission range* of a sensor s_i is a disk of radius (R_i), including its boundary, centered at ξ_i (the location of s_i), and defined by the point set, $D(R_i, \xi_i) = \{\xi \in \mathbf{IR}^2 : |\xi_i - \xi| \leq R_i\}$.

Neighbor Set. The *neighbor set* of a sensor (s_i) is given by $N(s_i) = \{s_j : |\xi_i - \xi_j| \leq R_j\}$, where R_i is the radius of the transmission range of s_i.

Coverage. Let A be an area of the field. A point $p \in A$ is said to be *covered* (or *sensed*) if and only if it belongs to the sensing range of at least one

sensor. The area A is said to be covered if and only if for every point $p \in A$ is covered.

Homogeneous versus Heterogeneous Network. A WSN is said to be *homogeneous* if all its sensors have the same storage, computation, communication, sensing, and energy capabilities. Otherwise, it is *heterogeneous*.

Communication Graph. A communication graph of a homogeneous (*heterogeneous*) WSN is an undirected (*directed*) graph, $G = (S, E)$, where S is a set of sensors and E is a set of (*directed*) edges between them such that for all $s_i, s_j \in S$, $(s_i, s_j) \in E$ if $|\xi_i - \xi_j| \leq R_i$.

Connectivity and Fault Tolerance. Let $G = (S, E)$ be a communication graph representing a network, where S is a set of sensors and E is a set of communication links between them such that for all $s_i, s_j \in S$, $(s_i, s_j) \in E$ if $|\xi_i - \xi_j| \leq R_i$. The *vertex-connectivity* (or *connectivity*) of G is equal to K if and only if G can be disconnected by the removal of at least K nodes. The *fault tolerance* of G is equal to K − 1.

Voronoi Diagram. Let $S = \{s_0, \ldots, s_{m-1}\}$ be a finite set of m sites in the plane. The *Voronoi diagram* [7] of S, denoted by $Vor(S)$, is a subdivision of the plane containing S into m *Voronoi* regions $VR(s_i)$, for $1 \leq i \leq m$, as shown in Fig. 4.1. Note that $VR(s_i)$ is possibly an unbounded open convex polygonal region that consists of all points closer to s_i than any other site in S. The edges of this region are called *Voronoi edges*. The $Vor(S)$ is the union of all *Voronoi* regions of sites $s_i \in S$. A WSN can be modeled by a *Voronoi* diagram with sites representing locations of sensors.

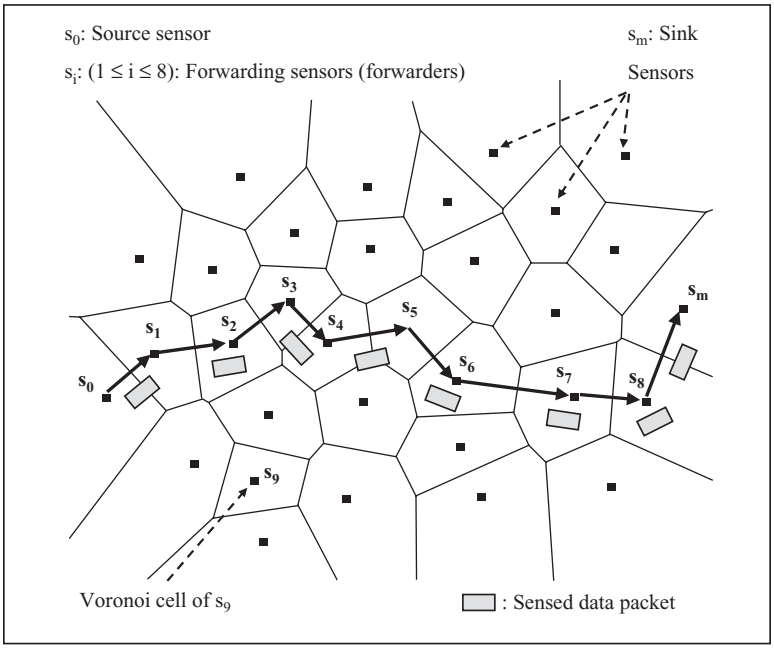

Fig. 4.1 The *Voronoi* diagram of a wireless sensor network.

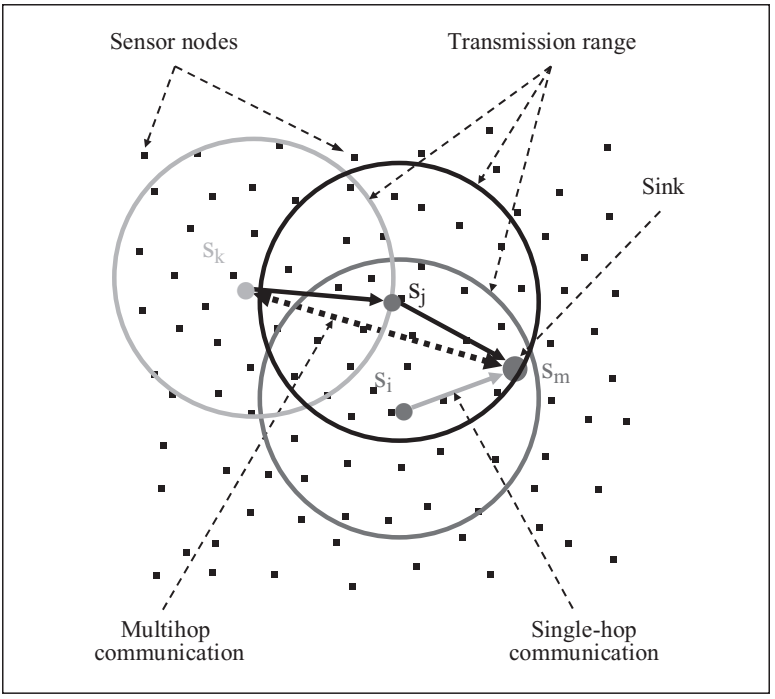

Fig. 4.2 Architecture of a wireless sensor network.

Figure 4.2 shows a network composed of a set of sensors randomly deployed in a square sensor field. The transmission range of a sensor is represented by a circle. When a sensor needs to communicate with another sensor that is inside its transmission range, the communication can be *single hop* (or direct). Otherwise, it must be *multihop* (or indirect) via other intermediate sensors that act as relays between the two communicating sensors. While the sensor s_i can communicate directly with the sink s_m, the sensor s_k can communicate with s_m only through other intermediate sensors, for example, s_j.

4.2.1.2 Energy Model. We assume that the energy consumption of the sensors is due to data transmission and reception. According to Ref. [6], the energy consumed in transmitting one message of size κ bits over a distance d called *transmission distance*, is given by $E_{tx}(d) = (\varepsilon d^{\alpha} + E_{elec})\kappa$, where E_{elec} represents the electronic energy, $\varepsilon \in \{\varepsilon_{fs}, \varepsilon_{mp}\}$ is the transmitter amplifier in the free space (ε_{fs}) or the multipath (ε_{mp}) model, and α is the path-loss exponent, $2 \le \alpha \le 4$. Also, the energy consumed in message reception is given by $E_{rx} = \kappa E_{elec}$. Hence, the total energy consumption when a sensor receives a message and forward it over a distance d is given by $E_{tot}(d) = (\varepsilon d^{\alpha} + 2E_{elec})\kappa$.

4.2.2 Challenges

The design of routing and data dissemination protocols for WSNs is challenging because of several network constraints. These constraints are imposed not only by the characteristics of individual sensors, the behavior of a network, and the nature of sensor fields, but also by the requirements of a sensing application in terms of some desirable metrics.

4.2.2.1 Sensor Characteristics. WSNs suffer from the limitations of several network resources, for example, energy, bandwidth, central processing unit (CPU), and storage [3], where energy is the most crucial resource because it determines the lifetime of a sensor. Also, energy poses a big challenge for network designers especially in hostile environments, for example, a battlefield, where it is impossible to access the sensors and recharge their batteries. Furthermore, when the energy of a sensor reaches a certain threshold, the sensor will become faulty and will not be able to function properly, which will have a major impact on the network performance. Therefore, algorithms designed for sensors should be as energy efficient as possible to extend their lifetime, and hence prolong the network lifetime while guaranteeing good performance overall.

Another challenge that faces the design of routing and data dissemination protocols is to manage the locations of the sensors. Most of the proposed protocols assume that the sensors either are equipped with *global positioning system* (GPS) receivers or use some localization technique [8] to learn about their locations. On one hand, although high sensor-location accuracy could be achieved, it is not cost effective that each sensor is equipped with a GPS receiver given that a WSN is highly dense in nature. On the other hand, the use of a localization technique may introduce certain inaccuracy in estimating the locations of the sensors.

4.2.2.2 Field Nature. As mentioned earlier, a sensor field may cause a difficulty not only in accessing the sensors for replacing and/or recharging their batteries, but also in their deployment. Thus, a deterministic sensor deployment strategy is not always possible. Such a strategy would help cover the field appropriately and minimize the total number of sensors required to achieve the specific requirements of sensing applications in terms of their expected type of coverage. In the real world, an application may require partial coverage, where only a certain percentage of the field is covered; full coverage, where the entire field is covered; or redundant coverage, where every location in the field is covered by multiple sensors simultaneously. In the case where sensors cannot be deployed deterministically because of the field nature, random deployment is the only remaining strategy. With random deployment, however, there is no guarantee that the coverage required by an application would be satisfied. There may be some areas that are not covered well or even not covered at all, which would lead to a problem known as *coverage hole*. Moreover, all deployed sensors are not guaranteed to be connected to each other or to the sink. This situation would lead to

another problem known as *connectivity hole*. These are two reasons that in most cases WSNs are designed with densely deployed sensors. Thus, the nature of a field has an influence on the network and this is a challenge for the designers and the investing party. As discussed later, one of the most widely used assumptions in the design of routing and data dissemination protocols is a high density of sensors deployed in a network. Although a highly densely deployed network needs more than necessary sensors, it helps guarantee network connectivity and achieve the coverage required by an application.

4.2.2.3 Network Characteristics. The topology of a network, which is defined by the sensors and the communication links between the sensors, changes frequently due to sensor addition and deletion. When a new sensor decides to join the network, the neighbor set of some sensors have to be updated. In many cases, it is necessary to add more sensors to maintain certain coverage properties of a sensor field and network connectivity. Similarly, when sensors deplete all their energy, they are considered faulty and no longer belong to the network. In this case, the neighbor sets of the faulty sensors should be updated as well. Also, in a mobile network, the network topology gets updated as sensors move in the sensor field. Consequently, any topology change in the network will have an influence on the communication paths (or routes) between the sensors. Therefore, routing and data dissemination paths should consider network topology dynamics due to limited energy and sensor mobility as well as increasing the size of the network to maintain specific application requirements in terms of coverage and connectivity. In particular, connectivity to the sink is very important. If the sensed data cannot reach the sink or there is no communication path between the source sensors (or data generators) and the sink, maintaining coverage would become not meaningful. Therefore, connectivity between all source sensors and the sink, either directly or indirectly, should be guaranteed for the proper operation of the network.

Another challenge is network scalability. In other words, routing and data dissemination protocols should be able to scale with the network size. Also, sensors may not necessarily have the same capabilities in terms of energy, processing, sensing, and particularly communication. Hence, communication links between sensors may not be symmetric, that is, a pair of sensors may not be able to have communication in both directions. This should be taken care of in the routing and data dissemination protocols.

4.2.2.4 Sensing Application Requirements. In most sensing applications, the sensed data should be as accurate as possible to assure better decision making by the sink. Moreover, the sensed data should reach the sink in a timely manner. Also, data redundancy is sometimes desirable in that it increases data accuracy. For example, in the intruder detection and tracking application, multiple sensors should be active in order to gather enough information about the intruder and track its motion accurately. Therefore, the routing and data dissemination protocols should guarantee data delivery and its accuracy so that the sink

can gather the required knowledge about the physical phenomenon on time. Furthermore, sensors may deplete their energy and become faulty. As discussed earlier, the sensor field may not be accessible and thus replacing those faulty sensors would be impossible. Hence, a network should tolerate the presence of faulty sensors and remain functional in spite of those faulty sensors. The degree of fault tolerance of the network depends on the underlying sensing application. Therefore, the routing and data dissemination protocols should also be fault tolerant.

4.3 TAXONOMY OF ROUTING AND DATA DISSEMINATION PROTOCOLS

This section presents a taxonomy of routing and data dissemination protocols for WSNs, as shown in Fig. 4.3. This taxonomy is based on several classification criteria, including location information, network layering and in-network processing, data centricity, path redundancy, network dynamics, quality-of-service (QoS) requirements, and network heterogeneity.

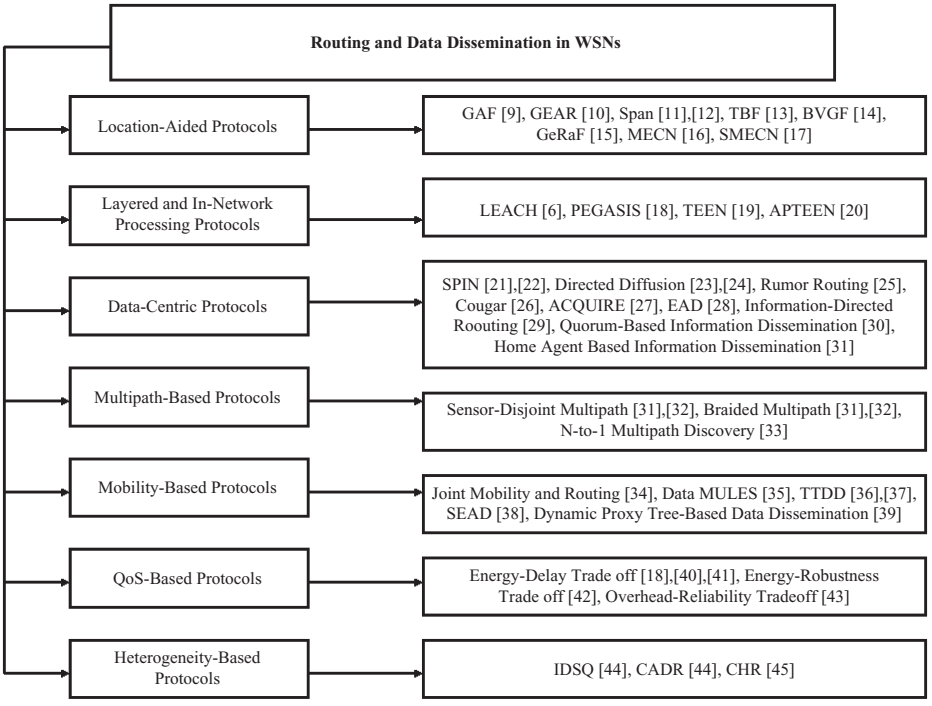

Fig. 4.3 Taxonomy of routing and data dissemination protocols for WSNs.

4.3.1 Location Information

The notion of physical location is an essential metric in several routing and data dissemination protocols in WSNs. Based on the location information of the sensors, these protocols can be short- or long range, depending on whether the distance between consecutive forwarders is minimum or maximum. The energy consumed in data forwarding depends on the distance over which data is transmitted. Note that location information was first used by routing protocols for mobile ad hoc networks (MANETs) [46]. While energy is not a metric in some MANET routing protocols, for example, location-aided routing (LAR) [46], it should be considered in the design of routing protocols for WSNs.

4.3.2 Network Layering and In-Network Processing

The architecture of a network can be flat in the sense that all sensors have the same role. In other words, all sensors forward their sensed data to the sink without necessarily passing through a particular node. A network is said to be *nonlayered* if all its sensors form only one group in which the sensors collaborate together to accomplish a common monitoring task. On the other hand, the sensors in a network can be grouped into *clusters*, each of which is managed by a specific sensor called a *cluster head*. These types of networks are said to be *layered*, where any sensed data should pass through one or more cluster heads before reaching the sink. These cluster heads are supposed to be powerful enough so that they can process the data they receive before sending them to the sink. All other sensors only need to sense the environment and send their data to the cluster heads for further processing. In some sensing applications, redundancy and correlation exist in the gathered data. Hence, it would be desirable to transmit only more representative data. For example, in monitoring the temperature of a room, the variation of the data within a given region is expected to be small. Thus, the sink is not interested in receiving all the temperature measures, but rather only some of them. This would reduce the communication overhead introduced by data forwarding significantly and improve the network performance. Also, the concept of layering makes a network more scalable and leads to more efficient usage of the energy of sensors, thus extending the network lifetime.

Extending network lifetime is an ultimate goal in the design of a WSN. Given that most energy of a sensor is mainly consumed in processing, sensing, and communication, an efficient design approach should take into account these three components of energy consumption. A question that network designers are mostly concerned about is *How can the lifetime of a network be extended?* To address this problem, several energy-efficient routing and data dissemination protocols have been proposed, which focus on how to forward the data until they reach the sink regardless of the type of data being transmitted from the source sensors to the sink. Among those protocols, one class does not update the data at intermediate sensors. That is, each intermediate sensor only acts as a pure data relay without altering any of the data it has received. Another class of protocols

introduces the concept of *in-network processing* to handle unnecessary redundancy and correlation contained in the sensed data. In many applications, the data sensed by the sensors have a certain amount of redundancy and correlation. It would be desirable if the sink can only receive relevant data for faster and better decision making. For this purpose, the sensed data should be processed at intermediate sensors before they reach the sink. The benefit of this in-network processing, such as data fusion, can be seen when vectorial data rather than scalar data are being transmitted. For example, in an application monitoring the temperature of a room, the sensed data are scalar (i.e., integer or real values). Hence, the cost of data communication is not very high, and the data fusion or aggregation is not costly as well. But sending continuously unnecessary redundant data will consume a huge amount of energy. If a sensing application has to send a large size of data, for example images, to the sink for further analysis and processing, it would consume a huge amount of energy. In this case, it would be more beneficial if those images sensed by different sensors could be aggregated and only a few of them would be sent. However, it is also true that processing those images for data fusion requires a considerable amount of energy. Moreover, there will be a delay due to the processing of those images. Therefore, there is a trade-off between data communication and fusion in this type of information intensive networks, where the sensed data are not scalar, but rather vectorial.

4.3.3 Data Centricity

A new communication paradigm has emerged in WSNs, which makes sensors capable of sensing, storage, processing, and computation to coordinate their sensing activities. This communication paradigm is *data centric* as all communications between sensors concern named data [47]. Because of its high density and mission nature, a WSN should be designed differently from IP-style networks in order to guarantee more efficient routing and data dissemination. Unlike general communication networks, a WSN is *task specific* in that a task to be performed by sensors is known at the time of sensor deployment.

4.3.4 Path Redundancy

In addition to their scalability and energy efficiency, the design of routing and data dissemination protocols for WSNs should also consider robustness, which means that a network remains functional in spite of the occurrence of sensor and link failures. Multipath routing is one technique that can make routing and data dissemination robust. This routing technique implies the existence of multiple paths between source and destination sensors (the sink is one of the destinations). These paths could be either disjoint or partially disjoint. Although maintaining alternate paths introduces some overhead and consumes more energy, multipath routing is an effective technique to improve robustness in the face of path failures that are caused by frequent topological changes due to unreliable wireless communication links and sensor failures. More specifically, multipath routing

helps recover from sensor and link failures and provide necessary resilience to the network at the cost of excessive redundancy.

4.3.5 Network Dynamics

As mentioned earlier, several factors, for example, limited energy and mobility, have an impact on the network topology. However, we consider energy a constraint, but not a goal. Any protocol designed for sensors should be as energy efficient as possible in order to address the constraint imposed by the limited energy of sensors. On the other hand, mobility is a desirable feature that can be used to tackle some problems in WSNs, for example, coverage hole and connectivity hole. In this taxonomy, we focus on mobility because it is the main source of network dynamics [48]. At the end, the sensed data will be transmitted over some established paths between the source sensors and the sink. The existence of these paths depends on whether the sensors are static or mobile. Thus, we classify the routing and data dissemination protocols based on whether a network is static or dynamic.

In a static network, there is no mobility at all; that is, both the sensors and the sink remain in their fixed locations during their collaborative mission of monitoring a physical environment. Therefore, there is not much overhead required to maintain routes between the sensors and the sink and between the sensors themselves. Actually, the locations of the sensors and the sink can be learned at the beginning of their monitoring task by exchanging some control messages. The neighbors of a given sensor are always the same unless a new sensor has joined the network or an existing sensor has left the network either by its will or because its entire energy is depleted.

In a mobile network, either the sensors are moving or the sink is moving. In any case, the routes between the sensors and the sink change frequently. A route that is currently valid might not be valid later on. This route instability would introduce an additional overhead for finding valid routes for data transmission and forwarding. As a result, the network may suffer from a delay in relaying the sensed data to the sink. In some scenarios, for example, the data MULES based architecture in [35], both the sensors and the sink are static, but there are other nodes acting as *relays*, which move in the sensor field to collect the sensed data from the source sensors and report them to the sink.

It is worth noting that whether mobility needs to be considered depends on the sensing application. For example, if we are interested in controlling the temperature, humidity, sound, or light in a room, there is no need to have mobile sensors or a mobile sink. However, for monitoring a moving object, it is necessary to introduce some degree of mobility to the network for an efficient tracking of the object. It has been proved that the use of mobile relays helps increase the lifetime of a WSN [49].

4.3.6 Quality of Service Requirements

Sensing applications may have different requirements, which can be expressed in terms of some QoS metrics, such as delay, reliability, and fault tolerance.

For example, time-critical applications have delay bounds to meet. For such applications, the sensed data must reach the sink within a certain time. Also, a desired property of sensing applications is fault tolerance by which it is meant that a network should remain functional in the event of sensor failures. Another desired property is reliability by which it is meant that the sensed data should be received by the sink as correctly as possible to ensure accurate decision making by the sink. Both fault tolerance and reliability require the deployment of more than necessary sensors so that the network can continue to function properly and deliver accurate sensed data to the sink despite some sensor failures. However, the use of redundant sensors yields additional energy consumption. Therefore, routing and data dissemination protocols should be designed in a way to trade-off between energy, fault tolerance, reliability, and delay. Recall that energy is a constraint that should be met by any routing and data dissemination protocol in order to guarantee an efficient usage of the amount of energy available at each sensor.

4.3.7 Network Heterogeneity

Most of the protocols designed for WSNs assume that the sensors have the same capabilities in terms of storage, processing, sensing, and communication. The resulting network is said to be *homogeneous*, where all communication links between the sensors are symmetric, that is, a given pair of neighboring sensors can directly communicate with each other. In these types of networks, a pair of sensors would have the same lifetime if they have the same energy consumption rate. Some sensing applications, however, use sensors with different capabilities and accordingly the resulting network is said to be *heterogeneous*. In the real world, the assumption of homogeneous sensors may not be practical because sensing applications may require heterogeneous sensors in terms of their sensing and communication capabilities in order to enhance network reliability and extend network lifetime [50]. Also, even if the sensors are equipped with identical hardware, they may not always have the same communication and sensing models. In fact, at the manufacturing stage, there is no guarantee that two sensors using the same platform have exactly the same physical properties. This taxonomy focuses on heterogeneity at the designing stage, when sensors are designed to have nonidentical capabilities to meet the specific needs of sensing applications.

4.4 OVERVIEW OF ROUTING AND DATA DISSEMINATION PROTOCOLS

Traditional routing protocols have several shortcomings when applied to WSNs, which are mainly due to the energy-constrained nature of such networks. For example, *flooding* is a technique in which a given node broadcasts data and control packets that it has received to the rest of the nodes in the network. This

process repeats until the destination node is reached. Note that this technique does not take into account the energy constraint imposed by WSNs. As a result, when used for data routing in WSNs, it leads to the following two problems, namely, *implosion* and *overlap* [4]. Given that flooding is a blind technique, duplicated packets may keep circulate in the network, and hence sensors will receive those duplicated packets, causing an implosion problem. Also, when two sensors sense the same region and broadcast their sensed data at the same time, their neighbors will receive duplicated packets. To overcome the shortcomings of flooding, another technique known as *gossiping* can be applied. In *gossiping*, upon receiving a packet, a sensor would select randomly one of its neighbors and send the packet to it. The same process repeats until all sensors receive this packet. Using gossiping, a given sensor would receive only one copy of a packet being sent. While gossiping tackles the implosion problem, there is a significant delay for a packet to reach all sensors in a network.

This section surveys a sample of existing routing and data dissemination protocols for WSNs and classifies them with respect to the taxonomy introduced in Section 4.3.

4.4.1 Location-Aided Protocols

There are several location-based routing protocols proposed for WSNs, for example, *greedy other adaptive face routing* (GOAFR) [51], *greedy perimeter stateless routing* (GPSR) [52], *most forward with fixed radius* (MFR) [53], *geographic distance routing* (GEDIR) [54], to name a few. However, those protocols were initially designed for MANETs without any energy considerations. They do not consider the specific requirements of WSNs, particularly their limited energy resources, and therefore cannot be used for such networks.

This section presents a sample of location-aware routing and data dissemination protocols proposed for WSNs, as well as some of those proposed for MANETs with energy consideration. Both types of protocols do not use flooding due to the implosion and overlap problems it can cause.

4.4.1.1 Geographic Adaptive Fidelity. *Geographical adaptive fidelity* (GAF) [9] is a routing protocol proposed for MANETs. Although it was proposed for MANETs, it favors energy conservation and thus can be used for WSNs. Hence, we will use the word *sensor* instead of *node*, which is used in GAF. The design of GAF is motivated by the results of the previous studies based on an energy model that considers energy consumption due to the reception and transmission of packets as well as idle (or listening) time when the radio of a sensor is on to detect the presence of incoming packets. These studies [48,55] showed that battery-powered nodes consume energy not only when receiving or sending packets, but also when listening or idle. Therefore, it is not enough to optimize energy consumption by only reducing packet transmission and reception. In addition, the radio should also be turned off. GAF is based on this mechanism; that is, turning off unnecessary sensors while keeping a constant

level of *routing fidelity* (or uninterrupted connectivity between communicating sensors).

GAF divides a sensor field into grid squares and every sensor uses its location information, which can be provided by GPS or other location systems [8,56,57], to associate itself with a particular grid in which it resides. This kind of association is exploited by GAF to identify the sensors that are equivalent from the perspective of packet forwarding. The size of the grid square is chosen in a way such that sensors within the same grid are equivalent with regard to routing and that sensors in adjacent grids can communicate with each other. Thus, equivalent sensors can coordinate with each other to determine an energy-efficient schedule of their activities, which specifies when and for how long the sensors stay awake or sleep.

As shown in Fig. 4.4, the state transition diagram of GAF has three states, namely, *discovery, active*, and *sleeping*. When a sensor enters the *sleeping* state, it turns off its radio for energy savings. In the *discovery* state, a sensor exchanges discovery messages to learn about other sensors in the same grid. Even in the *active* state, a sensor periodically broadcasts its discovery message to inform equivalent sensors about its state. The time spent in each of these states can be tuned by the application depending on several factors, such as its needs and sensor mobility. GAF aims to maximize the network lifetime by reaching a state where each grid has only one active sensor based on sensor ranking rules. The ranking of sensors is based on their residual energy levels. Thus, a sensor with a higher rank will be able to handle routing within their corresponding grids. For example, a sensor in the *active* state has a higher rank than a sensor in the *discovery* state. A sensor with longer expected lifetime has a higher rank.

In order to have all the sensors running for as long as possible without penalizing any one of them, GAF uses a load balancing strategy in which a sensor remains in the *active* state for only some time before switching to the *sleeping*

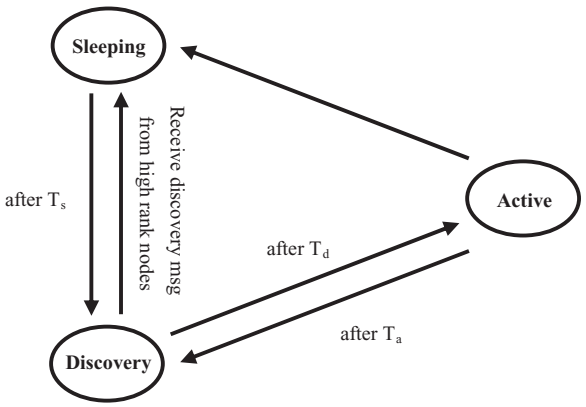

Fig. 4.4 State transition diagram of GAF.

state. This would give a chance to other sensors within the same grid to become active and handle routing. The rationale behind this rule is that sensors switching to the *discovery* state would have less residual energy than their neighbors in the *sleeping* state, where they conserve their energy. Note that sensor mobility may leave a grid with no active sensors at all. To address this problem, a sensor estimates the time it expects to leave its grid based on its GPS receiver and advertises it in its discovery message. Upon receiving this discovery message, the sensor's neighbors adjust their sleeping time so that their grid has always one active sensor to handle routing within that grid.

4.4.1.2 Geographic and Energy-Aware Routing. Yu et al. [10] proposed an energy-efficient routing protocol, called *geographic and energy aware routing* (GEAR), for routing queries to target regions in a sensor field. In GEAR, the sensors are supposed to have localization hardware equipped, for example, a GPS unit or a localization system [8] so that they know their current positions. Furthermore, the sensors are aware of their residual energy as well as the locations and residual energy of each of their neighbors. GEAR uses energy aware heuristics that are based on geographical information to select sensors to route a packet toward its destination region. Then, GEAR uses a recursive geographic forwarding algorithm to disseminate the packet inside the target region. The goal behind using energy aware data dissemination with geographical information is to help make energy-efficient routing decisions. GEAR is motivated by the fact that in several location-aware systems, such as WSNs, it is useful to disseminate information to a geographical region. For example, a user could interrogate the sensing application about the temperature in a given region within some time interval. To receive an answer, this query should be disseminated to the sensors located in the target region. The location information added to the query will help it to be sent directly to its ultimate destination area rather than flooding it in the entire sensor field.

For each of its neighbors, a sensor maintains two variables, called *estimated cost* and *learned cost*. The estimated cost of a neighbor N_i depends on the consumed energy at N_i and the distance between N_i and the centroid of the target region. If a sensor does not have the learned cost for its neighbor N_i, it computes the estimated cost as a default value for the learned cost. A sensor selects the neighbor $N_{\min}$ with minimum learned cost in order to balance the energy consumption across all its neighbors. After the selection process, a sensor sets its own learned cost to the sum of the learned cost of $N_{\min}$ and the cost of transmitting a packet to $N_{\min}$.

GEAR has mainly two phases, namely, forwarding a packet toward its destination region (phase 1) and disseminating the packet within the destination region (phase 2). During phase 1, a sensor selects a neighbor that is closer to the destination region than itself to act as the next forwarder. Otherwise, all its neighbors are farther away from the destination region than itself, and hence there is a void region between the sensor holding a packet and the target region (see Fig. 4.5). In this case, GEAR selects one of those neighbors whose learned

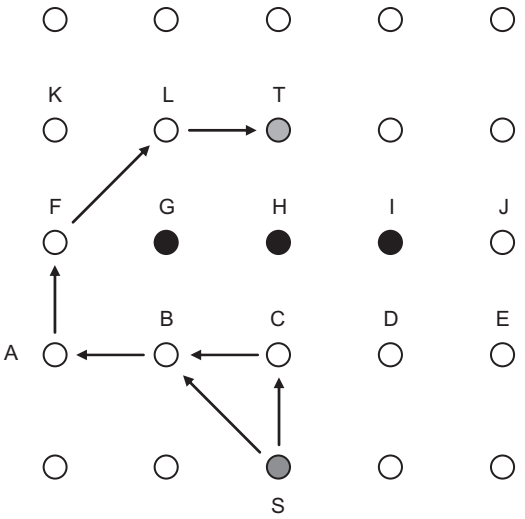

Fig. 4.5 Routing while avoiding holes.

cost is the minimum. In phase 2, GEAR uses a recursive geographic forwarding algorithm to disseminate the packet within the target region. In this case, the target region is split into four subregions and the current sensor creates four copies of the packet to be unicast to those subregions. This procedure of splitting and forwarding repeats until the current node finds itself to be the only one inside this subregion, and hence the packet is dropped. When the sensors are sparsely deployed, GEAR uses restricted flooding, which is more energy efficient than recursive geographic forwarding. In this case, a sensor sends only one broadcast message to all its neighbors.

Note that GEAR can also be classified as a data-centric data dissemination protocol, which will be discussed in Section 4.4.8. In GEAR, a query is expressed in terms of the name of the data, for example, temperature, not the sensor identifiers.

4.4.1.3 Coordination of Power Saving with Routing. *Coordination of power saving with routing* (Span) [11,12] is a routing protocol proposed for MANETs, but can be applied to WSNs as its goal is to reduce energy consumption of the nodes. In the context of WSNs, we also use a sensor to refer to a node in Span. Span is motivated by the fact that the wireless network interface of a device is often the single largest consumer of power. Hence, it would be better to turn the radio off during idle time. Although Span does not require that sensors know their location information, it runs well with a geographic forwarding protocol. Span helps sensors to join a forwarding backbone topology as coordinators that will forward packets on behalf of other sensors between any source and destination. According to Span, a sensor is eligible for becoming a coordinator if any pair of its neighbors cannot communicate either directly or via at most two

coordinators. To give a chance to other noncoordinator sensors to become coordinators, a coordinator withdraws if any pair of its neighbors can reach each other via some neighbors even if those neighbors are not currently coordinators. When used with a geographic forwarding protocol, Span's election rule requires each sensor to advertise its status (i.e., coordinator or noncoordinator), its neighbors, and its coordinators. Furthermore, when it receives a packet, a coordinator forwards the packet to a neighboring coordinator if any, which is the closest to the destination or to a noncoordinator that is closer to the destination.

4.4.1.4 Trajectory-Based Forwarding. *Trajectory-based forwarding* (TBF) [13] is another protocol that can be used in dense ad hoc networks, for example, WSNs. TBF requires a sufficiently dense network and the presence of a coordinate system, for example, a GPS, so that the sensors can position themselves and estimate distance to their neighbors. This new paradigm benefits from the characteristics of source-based routing, for example, *dynamic source routing* (DSR) and *Cartesian routing*. The source specifies the trajectory in a packet, but does not explicitly indicate the path on a hop-by-hop basis. The trajectory is expressed in the parametric form $\{X(t), Y(t)\}$ that is suitable for the purpose of forwarding, where t is a parameter, for example, the distance along the curve, indicated by the source. In such a representation of the trajectory, the parameter of the curve is a proxy for the hop count and represents a metric that measures the forward progress along the path. Moreover, a trajectory can be composed of several simple trajectories, each of which can be considered as a segment that can be represented by an appropriate interval of the parameter associated with the trajectory. Furthermore, the forwarding nodes are selected based on their proximity to the trajectory, not to the destination. Based on the location information of its neighbors, a forwarding sensor makes a greedy decision to determine the next hop that is the closest to the trajectory fixed by the source sensor. In fact, the specification and evaluation of a trajectory has certain cost in terms of complexity.

As can be seen, route maintenance in TBF is unaffected by sensor mobility given that a source route is a trajectory that does not include the names of the forwarding sensors. In order to increase the reliability and capacity of the network, it is also possible to implement multipath routing in TBF where an alternate path is just another trajectory. Note that it is not necessary to specify a final destination in the trajectory. This helps implement some networking functions, for example, flooding, discovery, and network management. To flood a packet in the network, a source sensor could specify the directions and lengths of radial lines in order to provide a satisfactory coverage of the sensors in the network. In addition to radial lines, trajectories could be specified as H-trees or fractals to achieve the required coverage. As mentioned earlier, the trajectory specification and evaluation has some complexity and it would be beneficial to the trade-off between this complexity and the required coverage. TBF can also be used for resource discovery. For example, a server could advertise its location along an arbitrary trajectory (or line) and a client could send its query along another trajectory that

will eventually intersect the server's trajectory. The sensor located at the intersection point will then inform the client about the position of the server. The client can then transmit its request along another trajectory to the server. Another interesting application of TBF is securing the perimeter of the network. For this purpose, a source sensor could specify a trajectory as a boomerang, where a packet including a challenge response token is sent along this trajectory. Any sensor that answers properly can be considered as authenticated.

4.4.1.5 Bounded Voronoi Greedy Forwarding. Bounded *Voronoi greedy forwarding* (BVGF) [14] is another location-based routing protocol for WSNs, which uses the concept of *Voronoi* diagram [5]. Therefore, the sensors should be aware of their geographical positions. In BVGF, a network is modeled by a *Voronoi* diagram with sites representing the locations of sensors. In this type of greedy geographic routing, a sensor will always forward a packet to the neighbor that has the shortest distance to the destination. The sensors eligible for acting as the next hops are the ones whose *Voronoi* regions are traversed by the segment line joining the source and the destination. The BVGF protocol chooses as the next hop the neighbor that has the shortest Euclidean distance to the destination among all eligible neighbors. It does not help the sensors deplete their battery power uniformly. Each sensor actually has only one next hop to forward its data to the sink. Therefore, any data dissemination path between a source sensor and the sink will always have the same chain of the next hops, which will severely suffer from battery power depletion. BVGF does not consider energy as a metric.

4.4.1.6 Geographic Random Forwarding. A relay sensor in data forwarding toward the sink is usually referred to as a sender. Zorzi and Rao [15] proposed a new data transmission protocol, called *geographic random forwarding* (GeRaF), which uses geographic routing where a sensor acting as relay is not known *a priori* by a sender. As will be discussed below, there is no guarantee that a sender will always be able to forward the message toward its ultimate destination, that is, the sink. This is the reason that GeRaF is said to be *best-effort* forwarding. GeRaF assumes that all sensors are aware of their physical locations, as well as that of the sink. Although GeRaF integrates a geographical routing algorithm and an awake–sleep scheduling algorithm, the sensors are not required to keep track of the locations of their neighbors and their awake–sleep schedules.

When a source sensor has sensed data to send to the sink, it first checks whether the channel is free in order to avoid collisions. If the channel remains idle for some period of time, the source sensor broadcasts a request-to-send (RTS) message to all of its active (or listening) neighbors. This message includes the location of the source and that of the sink. Note that the coverage area facing the sink, called *forwarding area*, is split into a set of N_p regions of different priorities such that all points in a region with a higher priority are closer to the sink than any point in a region with a lower priority. When active neighboring sensors

receive the RTS message, they assess their priorities based on their locations and that of the sink. The source sensor waits for a CTS message from one of the sensors located in the highest priority region. For GeRaF, the best relay sensor is the one closest to the sink, thus making the largest advancement of the data packet toward the sink. In case that the source does not receive the CTS message, it implies that the highest priority region is empty. Hence, it sends out another RTS polling sensors in the second highest priority region. This process continues until the source receives the CTS message, which means that a relay sensor has been found. Then, the source sends its data packet to the selected relay sensor, which in turn replies back with an ACK message. The relay sensor will act in the same way as the source sensor in order to find the second relay sensor. The same procedure repeats until the sink receives the sensed data packet originated from the source sensor. It may happen that the sending sensor (source sensor or relay sensor) does not receive any CTS message after sending N_p RTS messages. This means that the neighbors of the sending sensor are not active. In this case, the sending sensor backs off for some time and retries later. After a certain number of attempts, the sending sensor either finds a relay sensor or discards the data packet if the maximum allowed number of attempts is reached.

A question that can arise is *Which sensors are allowed to reply to a given RTS message*? If a sensor is in the highest priority region and receives an RTS message, it replies immediately with a CTS message. In general, a sensor in the ith priority region replies with a CTS message only if it receives the ith RTS message and the first (i-1) RTS messages were not answered. In case that there are multiple CTS messages answering a given RTS message, some collision resolution algorithm is triggered by sending a special RTS message, which results in selecting only one relay sensor among all sensors located in the priority region being considered by the original RTS message.

The interested reader can also refer to [58] for a detailed description of the collision avoidance protocol, as well as a detailed analysis of the energy and latency performance of GeRaF.

4.4.1.7 Minimum Energy Communication Network. In all previous discussed protocols, static sensors are assumed. In Ref. [16], a location-based protocol for achieving minimum energy for randomly deployed ad hoc networks was proposed. This protocol, called *minimum energy communication network* (MECN), can be used for WSNs. MECN attempts to set up and maintain a minimum energy network with mobile sensors. The motivation of MECN is based on the key premise that maximizing the total battery lifetime of a network requires minimizing the energy consumption of the entire network. MECN is a self-reconfiguring protocol that maintains network connectivity in spite of sensor mobility. It computes an optimal spanning tree rooted at the sink, called *minimum power topology*, which contains only the minimum power paths from each sensor to the sink. It is based on the positions of sensors on the plane and consists of two main phases, namely, *enclosure graph construction* and *cost distribution*.

For a stationary network, in the first phase (i.e., *enclosure graph construction*), MECN constructs a sparse graph, called an *enclosure graph*, based on the immediate locality of the sensors. For this purpose, a sensor first determines its relay region with respect to each of the sensors it can communicate with directly. A relay region contains all the points where relaying a message to any of these points through an intermediate sensor (or relay sensor) is always more energy efficient than sending the message to them directly. This determines a region around a sensor, called an *enclosure region*, beyond which it is not energy efficient to search for more neighbors. The sensors located in the enclosure region of a sensor are its neighbors to which the sensor will maintain communication links for energy-efficient transmission. As can be seen, the enclosure region of a sensor is bounded by the intersection of all relay regions with respect to all the sensors it can interact with directly. An enclosure graph is a directed graph that includes all the sensors as its vertex set and whose edge set is the union of all edges between the sensors and the neighbors located in their enclosure regions. In other words, a sensor will not consider the sensors located in its relay regions as potential candidate forwarders of its sensed data to the sink. Figure 4.6 shows the relay region of a transmit-relay pair of sensors. Only the sensors in its immediate neighborhood (i.e., *enclosure region*) will be the only potential candidate forwarders. Furthermore, this graph is sparse and strongly connected.

In the second phase (i.e., *cost distribution*), nonoptimal links of the enclosure graph are simply eliminated and the resulting graph is a *minimum power topology*. This graph has a directed path from each sensor to the sink and consumes the least total power among all graphs having directed paths from each sensor to the sink. To find optimal links on the enclosure graph, the Bellman–Ford shortest path algorithm is applied using the power consumption as the cost metric.

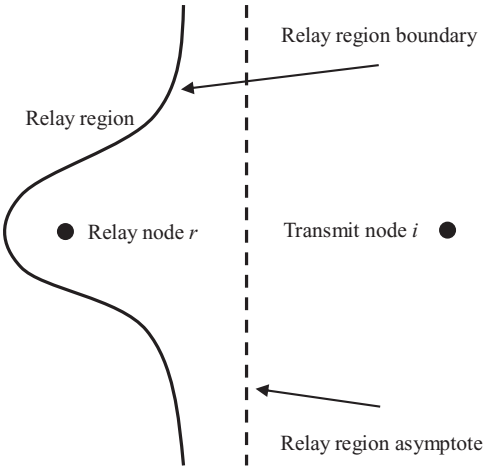

Fig. 4.6 Relay region of the transmit-relay pair (*i, r*).

Each sensor broadcasts its cost to its neighbors, where the cost of a node is the minimum power required for this sensor to establish a directed path to the sink. Specifically, a sensor first calculates

$$C_{i,n} = \text{Cost}(n) + P_{\text{tx}}(i,n) + P_{\text{rx}}(n),$$

where $P_{\text{tx}}(i, n)$ is the power needed to transmit from i to n, $P_{\text{rx}}(n)$ is the power required for n to receive from any transmitting sensor, and Cost(n) is the cost computed by the neighbor n (i.e., n is a neighbor of the sensor i). Then, the cost of sensor i is calculated as

$$\text{Cost}(i) = \min_{n \in N(i)} C_{i,n},$$

where $N(i)$ is the neighbor set of i based on the concept of the enclosure region. Figure 4.7 illustrates this concept. At the end of this phase, every sensor will have computed the minimum-cost neighbor link, which will be used to send its sensed data to the sink. All the links form the minimum power topology.

MECN applies also to synchronous mobile WSNs, in which a GPS can be used to provide absolute time information for synchronization. For energy-efficiency purposes, a sensor can move back and forth between a *listen* mode to listen for any change in the network topology due to sensor mobility and a *sleep* mode to conserve its energy. The *cycle period*, defined as the time between two successive wakeups, is very critical. A short cycle period introduces much overhead, and hence wastes energy due to computing costs that change very slowly, while a long cycle period will not reflect the exact costs to the sink. Therefore, a tradeoff between these two scenarios is required.

While MECN is a self-reconfiguring protocol, and hence is fault tolerant (in the case of mobile networks), it suffers from a severe battery depletion problem

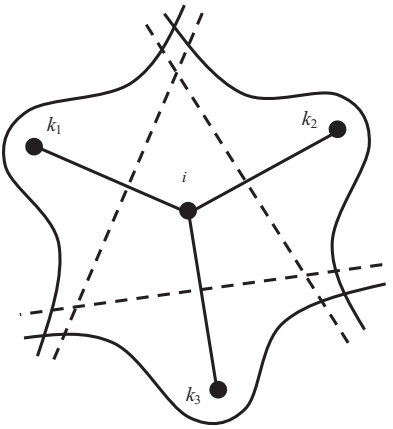

Fig. 4.7 Enclosure of sensor i.

when applied to static networks. The MECN does not take into consideration the available energy at each sensor, and hence the optimal cost links are static. In other words, a sensor will always use the same neighbor to transmit or forward sensed data to the sink. For this reason, this neighbor would die very quickly and the network thus becomes disconnected. To address this problem, the enclosure graph and thus the minimum power topology should be dynamic based on the residual energy of the sensors.

4.4.1.8 Small Minimum-Energy Communication Network. *Small minimum-energy communication network* (SMECN) is a protocol proposed to improve MECN discussed in Section 4.4.1.7. In this protocol, Li and Halpern [17] characterized a minimal graph with regard to the *minimum energy property*. This property implies that for any pair of sensors in a graph associated with a network, there is a minimum energy-efficient path between them; that is, a path that has the smallest cost in terms of energy consumption over all possible paths between this pair of sensors. Their characterization of a graph with respect to the minimum energy property is intuitive. A path between a pair of sensors u and v whose length is >1 is preferable to a direct edge (u, v) between them if the total power consumption of this path is less than that of the direct edge. A graph that satisfies this property is said to be *minimal* and is denoted by $G_{\min}$. In this case, the edge (u, v) is said to be k-redundant if the length of the energy-efficient path between u and v is equal to k.

The SMECN protocol attempts to construct a graph (i.e., a communication network) that includes $G_{\min}$ as a subgraph. For this purpose, SMECN needs to find a subset E_2 that contains all edges in the original graph that are not 2-redundant. Every sensor discovers its immediate neighbors by broadcasting a neighbor discovery message using some initial power that is updated incrementally. Specifically, the immediate neighbors of a given sensor are computed analytically. Then, a sensor starts broadcasting a neighbor discovery message with some initial power p and checks whether the theoretical set of immediate neighbors is a subset of the set of sensors that replied to that neighbor discovery message. If this is the case, the sensor will use the corresponding power p to communicate with its immediate neighbors. Otherwise, it increments p and rebroadcasts its neighbor discovery message. Li and Halpern [16] showed that E_2 is a subgraph of the enclosure graph produced by the MECN protocol.

4.4.2 Layered and In-Network Processing-Based Protocols

Traditional (or flat) routing and data dissemination protocols for WSNs may not be optimal in terms of energy consumption. Clustering is an energy-efficient communication protocol that can be used by the sensors to report their sensed data to the sink. In this section, we describe a sample of layered protocols in which a network is composed of several *clumps* (or *clusters*) of sensors. Each clump is managed by a special node, called *cluster head*, which is responsible for coordinating the data transmission activities of all sensors in its clump. All sensors

in a cluster communicate with the cluster head that acts as a local sink, which in turn transmits the sensed data to the global sink. Note that the transmission distance over which the sensors send their data to their cluster head is smaller compared to their respective distances to the global sink. Since a network is characterized by its limited wireless channel bandwidth, it would be beneficial if the amount of data transmitted to the sink can be reduced. To achieve this goal, a local collaboration between the sensors in a cluster is required in order to reduce bandwidth demands.

4.4.2.1 Low-Energy Adaptive Clustering Hierarchy. To overcome the shortcomings of conventional routing and data dissemination protocols, which run on top of nonlayered or flat network architectures, a clustering-based protocol, called *low-energy adaptive clustering hierarchy* (LEACH), was proposed in Ref. [6]. LEACH is based on an *aggregation* (or *fusion*) technique that combines or aggregates the original data into a smaller size of data that carry only meaningful information of all individual sensors. For this purpose, LEACH divides a network into several clusters of sensors, which are constructed by using localized coordination and control not only to reduce the amount of data that are transmitted to the sink, but also to make routing and data dissemination more scalable and robust. Given that the energy dissipation of the sensors depends on the distance and the data size to be transmitted, LEACH attempts to transmit data over short distances and reduce the number of transmission and reception operations.

In LEACH, the cluster heads are not selected in a static manner; otherwise, they will drain their energy and die quickly. Instead, LEACH uses a randomized rotation of the high-energy cluster-head position in order to give a chance to all sensors to act as cluster heads and avoid the battery depletion of an individual sensor. The operation of LEACH is divided into *rounds*, each of which has mainly two phases: a setup phase to organize the network into clusters and a steady-state phase for data transmission to the sink. Cluster heads use CSMA MAC protocol to advertise their status. Thus, all noncluster-head sensors must keep their receivers on during the setup phase in order to hear the advertisements sent by the cluster heads. These cluster heads are selected with some probability by themselves and broadcast their statuses to the other sensors in the network. The decision for a sensor to become a cluster head is made independently without any negotiation with the other sensors. Specifically, a sensor decides to become a cluster head based on the desired percentage P of cluster heads (determined *a priori*), the current round, and the set of sensors that have not become cluster heads in the past $1/P$ rounds. If the number of cluster heads is $<T(n)$, a sensor n becomes a cluster head for the current round, where $T(n)$ is a threshold given by

$$T(n) = \begin{cases} \dfrac{P}{1 - P(r \bmod 1/P)} & \text{if } n \in G, \\ 0 & \text{otherwise.} \end{cases}$$

The sensors that are cluster heads in round 0 cannot be a cluster for the next $1/P$-1 rounds. At round 0, each sensor has probability P to become a cluster head. Among all advertised cluster heads, a sensor selects the closest one that will incur minimum energy communication and then informs its cluster head about its decision to join the cluster using CSMA MAC protocol. Similarly, cluster heads should keep their receivers on to hear these join messages. Once the network is divided into clusters, a cluster head computes a TDMA schedule for its sensors specifying when a sensor in the cluster is allowed to send its data. Thus, a sensor will turn its radio on only when it is authorized to transmit according to the schedule established by its cluster head, thus yielding significant energy savings. Furthermore, LEACH enables data fusion in each cluster by aggregating the data in order to reduce the total amount of data before sending them to the sink. In another word, once a cluster head gathers all the data from its sensors, it aggregates them and transmits the aggregated data to the sink.

LEACH can be viewed as a hybrid approach using short- and long-range based data forwarding. The sensors within a cluster transmit their sensed data over short distances, whereas cluster heads communicate directly with the sink. While LEACH helps the sensors within their cluster dissipate their energy slowly, the cluster heads consume a larger amount of energy when they are located farther away from the sink. Sending directly to the sink is the main problem with LEACH. A better approach is to allow multihop data transmission to the sink through other cluster heads. In this case, a cluster head does not have to update the aggregated data from other cluster heads, but only forward them toward the sink. Moreover, a decision for a sensor to become a cluster head should consider the residual energy of that sensor. Table 4.1 shows that LEACH outperforms all other protocols, including a data transmission protocol called *Direct*, in which the sensors transmit directly to the sink, the minimum transmission energy (MTE) protocol, in which each data packet must go through n low-energy transmissions and n receptions, and the static clustering protocol, in which all cluster-heads are selected at once.

4.4.2.2 Power-Efficient Gathering in Sensor Information Systems. In LEACH, all cluster heads should broadcast their advertisements to all sensors in the network. In addition, all of them should transmit their aggregated data to the sink in each round. To improve LEACH, another protocol, called *power-efficient gathering in sensor information systems* (PEGASIS) [18], was proposed, which allows only one cluster head to transmit to the sink in each round. Moreover, a sensor has to transmit to its local neighbors in the data fusion phase instead of sending directly to its cluster head as in the case of LEACH. In PEGASIS, sensors are organized in a way to form a chain, which can be performed either by the sensors themselves using a greedy algorithm or by the sink, which has to broadcast the chain to all sensors in the network. The construction phase assumes that all the sensors have global knowledge about the network, particularly, the positions of the sensors, and uses a greedy approach. Specifically, it starts with the furthest sensor to the sink to guarantee that sensors farther away from the

TABLE 4.1 Lifetime with Different Sensor Initial Energy

Energy (Joule/Sensor)	Protocol	Round First Sensor Dies	Round Last Sensor Dies
0.25	Direct	55	117
	MTE	5	221
	Static clustering	41	67
	LEACH	394	665
0.5	Direct	109	234
	MTE	8	429
	Static clustering	80	110
	LEACH	932	1312
1	Direct	217	468
	MTE	15	843
	Static clustering	106	240
	LEACH	1848	2608

sink have close neighbors. When a sensor fails or dies due to low battery power, the chain is constructed using the same greedy approach by bypassing the failed sensor.

The chain has two end sensors and in each data fusion phase only one leader (i.e., a sensor responsible for transmitting the fused data to the sink) will transmit the fused data to the sink. Any other intermediate sensor will fuse the data received from its neighbor with its own data and transmit the fused data to its neighbor located closer to the sink than itself so that the fused data get forwarded toward the sink. Note that all sensors will participate in the data fusion except the end sensors unless they are leaders, which will transmit the fused data to the sink. The data fusion phase in each round requires that a leader send a control token to the end sensors of the chain, where the data transmission should start. At the end, the leader receives two fused data from both sides of the chain, fuses them with its own data, and transmits the final fused data to the sink.

Table 4.2 shows a comparison between LEACH, PEGASIS, and *Direct*, in which all sensors transmit directly to the sink. The results in the first half of the table correspond to a sensor field of size 50 m × 50 m, while those in the second half correspond to a 100 m × 100 m sensor field. We vary the initial energy of individual sensors (0.25, 0.5, and 1) and the percentage of sensors that die (1–100%). Note that the number of rounds increases with the initial energy of the sensors. Also, PEGASIS outperforms both LEACH and *Direct* for both sizes of the network.

4.4.2.3 Threshold Sensitive Energy Efficient Sensor Network Protocol.

A sensing application can be designed in a way where the sensors either sense and transmit their sensed data periodically to the sink or react immediately to any sudden change in the value of the sensed attribute. While in the first scenario, a network is said to be *proactive*, the second scenario corresponds to a *reactive*

TABLE 4.2 Comparison between PEGASIS, LEACH, and Direct

Energy (Joule/Sensor)	Protocol	1%	20%	50%	100%
0.25	Direct	54	62	76	117
	LEACH	402	480	523	635
	PEGASIS	788	1004	1041	1096
0.5	Direct	108	124	152	235
	LEACH	803	962	1036	1208
	PEGASIS	1578	2011	2082	2192
1.0	Direct	215	248	304	471
	LEACH	1610	1921	2055	2351
	PEGASIS	3159	4023	4165	4379
0.25	Direct	14	16	20	30
	LEACH	166	204	232	308
	PEGASIS	335	624	684	779
0.5	Direct	28	32	40	61
	LEACH	339	408	461	576
	PEGASIS	675	1250	1362	1544
1.0	Direct	56	64	80	122
	LEACH	690	812	911	1077
	PEGASIS	1346	2497	2720	3076

network. For time-critical applications, a reactive network is more suitable than a proactive network. In order to trade-off between energy efficiency, data accuracy, and response time dynamically, a communication protocol, called *threshold sensitive energy efficient sensor network protocol* (TEEN), was proposed in Ref. [33]. TEEN uses hierarchical clustering, which groups sensors into clusters with each led by a cluster head. The sensors within a cluster report their sensed data to their cluster head. The cluster head sends aggregated data to higher level cluster heads until the data reach the sink. Thus, TEEN is a clustering communication protocol that targets a reactive network and enables cluster heads to impose a constraint on when the sensors should report their sensed data. Each cluster head broadcasts to its members a value, called *hard threshold* (H_T), for the sensed attribute, beyond which a sensor should turn its transmitter on to report its sensed data to its cluster head. In addition, a cluster head broadcasts another value, called *soft threshold* (S_T), which indicates a small change in the value of the sensed attribute, which triggers a sensor to turn on its transmitter and send its sensed data to the cluster head.

The sensors within a cluster can be scheduled using TDMA or CDMA in order to avoid collisions in a cluster. However, this will introduce delay when reporting time-critical data to the sink. At the beginning of a sensing task, a sensor transmits its sensed data when its value is higher than the hard threshold specified by its cluster head. Moreover, a sensor stores the current value *SV* of

the sensed attribute. Based on the values of the hard and soft thresholds, a sensor transmits its sensed data only if its value is higher than H_T and the difference between this current value and the previously stored value *SV* is $\geq S_T$. When a sensor sends its sensed data, it updates *SV* with the current value of its sensed attribute.

As can be seen, the hard threshold helps the sensors to transmit only significant information while the soft threshold further reduces the number of transmissions for sensed data. Thus, the sensors will send only sensed data that are of interest to the end user based on the hard threshold value and the change with respect to the previously reported data, thus yielding more energy savings. However, both values of the hard and soft thresholds have an impact on TEEN. These values should be set very carefully to keep the sensors responsive by reporting sensed data to the sink. It may happen that for some value of the hard threshold, the sensors are not able to transmit at all. In this case, the cluster heads will not receive any data at all from their members. Also, the changes between the values of the currently sensed data and the previously reported one may not reach the soft threshold. In this case, the sensors will not report to their cluster heads. In either case, the cluster heads cannot know whether their members have died because of low energy or the above-mentioned conditions on the values of the hard and soft thresholds are not satisfied. Therefore, TEEN is not suitable for sensing applications which require sensors to report their data on a regular basis.

The simulation results reported in [19] show that TEEN performs much better than LEACH. Furthermore, TEEN using a soft threshold outperforms TEEN with a hard threshold as expected.

4.4.2.4 Adaptive Periodic TEEN. To overcome the above-mentioned shortcomings of TEEN, a new protocol, called *adaptive periodic TEEN* (APTEEN), which combines the best features of both TEEN (time-critical data) and LEACH (periodic sensed data transmission) was proposed in Ref. [34]. Therefore, APTEEN is a hybrid clustering-based routing protocol that allows the sensors to send their sensed data periodically and react to any sudden change in the value of the sensed attribute by reporting the corresponding values to their cluster heads. Similar to TEEN, APTEEN uses the concept of hierarchical clustering for energy efficient communication between source sensors and the sink. After the clusters are formed, a cluster head broadcasts the sensed attributes of interest, the hard and soft thresholds, a TDMA schedule that assigns a slot to each sensor, and a maximum time interval between two successive reports sent by a sensor, called *count time* (T_c). This count time is used when sensors have to report their sensed data periodically to the sink.

Contrary to other existing query routing protocols, APTEEN can handle three types of queries. Specifically, it can answer *historical queries* by extracting historical data associated with the events that occurred in the past. It can also respond to *one-time queries* that give a snapshot view of the network. Moreover, it can reply to *persistent queries* that allow monitoring the network within a time interval with respect to some sensed attributes.

It has been shown through extensive simulations that APTEEN guarantees lower energy dissipation and a larger number of sensors alive [20]. Compared to LEACH and TEEN, the performance of APTEEN in terms of energy consumption and network lifetime lies between those of LEACH and TEEN. While in LEACH sensors transmit their sensed data continuously to the sink, in APTEEN sensors transmit their sensed data based on the threshold values.

4.4.3 Data-Centric Protocols

In traditional routing protocols for WSNs, also known as *address-centric* protocols, when the sink sends out a query for collecting data, each source sensor that has the appropriate data responds by sending its data to the sink independently of all other sensors. Data-centric protocols differ from address-centric protocols in the manner that the data is sent from source sensors to the sink. In *data-centric* protocols, when the source sensors send their data to the sink, intermediate sensors can perform some form of aggregation on the data originating from multiple source sensors and send the aggregated data toward the sink. This process can result in energy savings because less transmissions are required to send the data from the sources to the sink. This section reviews a sample of data-centric routing and dissemination protocols for WSNs.

4.4.3.1 Sensor Protocols for Information via Negotiation. For some applications, for example, intruder detection, the design of a surveillance network that provides a way to replicate complete and global views of the physical environment across the entire network is necessary. This type of network helps disseminate critical pieces of information to all sensors in the network so that they become aware of any critical event that may occur. Moreover, it enhances the fault tolerance of the network and helps the network to continue to function normally in the presence of sensor failures. Thus, disseminating individual sensor observations to all sensors in the network should be performed as energy efficient as possible, where all sensors are considered as potential sinks. In light of this, a family of adaptive protocols, called *sensor protocols for information via negotiation* (SPIN) [21,22], is suggested. The SPIN protocols were designed in a way to improve classic flooding protocols and overcome the problems they may cause, for example, implosion and overlap, which were discussed earlier. In addition, flooding, when used, makes the sensors blindly consume their available resources. The SPIN protocols are resource aware and resource adaptive. The sensors running the SPIN protocols are able to compute the energy consumption required to compute, send, and receive data over the network. Thus, they can make informed decisions for efficient use of their own resources.

In data dissemination using SPIN, sensors are resource aware in the sense that they make their decisions based on their missions, the information they have about the environment and other sensors, and their computational, communication, and energy resources. Specifically, the SPIN protocols are based on two key mechanisms: *negotiation* and *resource adaptation*. In terms of negotiation, SPIN

enables the sensors to negotiate with each other before any data dissemination can occur in order to avoid injecting nonuseful and redundant information in the network. Therefore, the data observed by the sensors need to be named. For this purpose, SPIN uses *meta-data* as the descriptors of the data that the sensors want to disseminate. The notion of meta-data avoids the occurrence of overlap given that the sensors can name the interesting portion of the data they want to get. Note that the size of the meta-data should definitely be less than that of the corresponding sensor data. Also, meta-data have application-specific formats and introduce additional costs for their storage, retrieval, and management. For example, sensors covering disjoint areas may use their unique IDs as meta-data. Thus, the meta-data x stands for "all sensed data of sensor x". As can be observed, this negotiation process tackles the problems of implosion and overlap introduced by classic flooding protocols. Contrary to the flooding technique, each sensor is aware of its resource consumption with the help of its own *resource manager* that is probed by the application before any data processing or transmission. This helps the sensors to monitor and adapt to any change in their own energy resources. For example, based on its available energy, a sensor may not want to participate in data forwarding on behalf of other sensors, thereby extending its lifetime and the operating lifetime of the network. Thus, both negotiation and resource adaptation effectively address the above-mentioned problems caused by classic flooding protocols.

The family of SPIN protocols is motivated by the fact that routing decisions are best made using not only the knowledge about the network topology, but also the application data layout and the available resources at each sensor. Hence, it is interesting to integrate the concepts of data naming and routing in WSNs. There are two protocols in the SPIN family: SPIN-1 (or SPIN-PP) and SPIN-2 (or SPIN-EC) [22]. While SPIN-1 uses a negotiation mechanism to reduce the energy consumption of the sensors, SPIN-2 uses a resource-aware mechanism for more energy savings. Both protocols allow the sensors to exchange information about their sensed data, thus helping them to obtain the data they are interested in. SPIN-1 is a three-stage handshake protocol by which the sensors can disseminate their data. This protocol applies for those networks using point-to-point transmission media (or point-to-point networks), in which two sensors can communicate exclusively with each other without interfering with other sensors. For a sensor to communicate with all its neighbors, it has to communicate with each of them separately. Hence, the energy and time required for a sensor to communicate with n neighbors is n times the energy and time required for a sensor to communicate with one neighbor. SPIN-1 consists of two stages: advertisement, request, and data transmission. In the *advertisement* stage, a sensor advertises its data by sending an advertisement (ADV) message containing the meta-data of the data it wants to share with other sensors. In the *request* stage, if a sensor is interested in the actual data being advertised and does not possess all of the advertised data, it sends back a request (REQ) message to the source of the ADV message listing all of the data it wants to acquire. In the *data*

transmission stage, the source of the ADV message retrieves the requested data and replies to the REQ message with a DATA message that contains the actual data with a meta-data header. The sensor that has received the requested data could advertise them to its neighbors.

SPIN-BC [22] improves SPIN-PP by using one-to-many communication instead of many one-to-one communications. It is a three-stage handshake protocol for broadcast transmission media, where the sensors in a network communicate with each other using a single shared channel. A sensor can communicate with all of its neighbors using only one ADV message. However, if a sensor wants to advertise or receive data, it has to wait for the channel to be free before sending its messages.

SPIN-2 differs from SPIN-1 in that it takes into account the residual energy of sensors. If the sensors have plenty of energy, SPIN-2 is identical to SPIN-1, and hence has the same three stages. However, when a sensor has low residual energy, it controls its participation in a data dissemination process. Specifically, if a sensor finds out that it can participate in a data dissemination process without having its residual energy to become below some low-energy threshold, it sends out an REQ message for any advertised data it is interested in. Otherwise, the sensor will not send a REQ message for the advertised data because it does not have energy to send its request, receive the data, and process them. Note that a SPIN-2 sensor consumes energy when it receives ADV and REQ messages even when its residual energy is below a certain low-energy threshold. SPIN-2 does not prevent this energy consumption from occurring.

While the family of SPIN protocols applies to lossless networks, it can be slightly updated to apply to lossy or mobile networks. In addition to the conservative approach of SPIN-2, periodic readvertisement of ADV messages are necessary to address lost ADV messages. Similarly, a sensor can retransmit REQ messages to compensate for DATA messages that do not arrive because of some time out. The SPIN-RL [22] is an updated version of SPIN-BC that helps the sensors reliably disseminate their data in a lossy network.

4.4.3.2 Directed Diffusion. The main requirements of WSNs are energy efficiency, scalability, and robustness with conserving energy resources being the most crucial one. *Directed diffusion* [23,24] is a data-centric paradigm for sensor query dissemination and processing that meets the above requirements. Examples of such queries can have the following forms: *How many pedestrians observed in the geographical area X? or In what direction vehicle Y in region Z is moving?* Sensors can be used to detect pedestrians or vehicle movements, collaborate with each other to disambiguate the locations of pedestrians or the movement direction of vehicles, and report their, possibly aggregated, results to the sink. For example, sensors within region Z may coordinate to select the best estimate of the vehicle movement direction and send it back to the query originator. Specifically, a sensor transforms the sampled waveforms generated by a vehicle into *event descriptions* that include the sensor's location information, the direction of

vehicle movement, a codebook value (or event code) for the vehicle, a timestamp, the intensity of the signal, and a degree of confidence in its estimation. In directed diffusion, sensors name their generated data by attribute-value pairs. If a sensor wants to receive data, it sends *interests* for named data. When sent by a source sensor, the data can be cached or transformed by intermediate sensors, which in turn may initiate interests based on the data that were previously cached. In addition, the data sent by the source sensor may be aggregated by intermediate sensors before being forwarded to their destination. More importantly, interests, data dissemination, and data aggregation occur in directed diffusion in a *localized* manner via exchanging messages between neighboring sensors.

Directed diffusion has several key elements: *data naming, interests and gradients, data propagation*, and *reinforcement*. As mentioned earlier, a sensing task can be described by a list of attribute-value pairs. For example, consider the following task: In every *I* ms for the next *T* seconds, send me a location estimate of any four-legged animal in sub-region *R* of the sensor field. This animal tracking task can be described by the following data:

Type = four-legged animal // detect animal location
Interval = 20 ms //send back events every 20 ms
Duration = 10 seconds // … for the next 10 seconds
Rect = [–100, 100, 200, 400] // from sensors within rectangle

This task description is called an *interest*. We assume that an interest is injected into the network at the sink. Hence, the sink periodically broadcasts an interest for each active task specifying a low data rate, which has the following form:

Type = four-legged animalInterval = 1 s
Rect = [–100, 100, 200, 400]
Timestamp = 01:20:40
ExpiresAt = 01:30:40

To ensure reliable interest transmission, an interest has a timestamp that allows the sink to refresh the interest by resending it with a monotonically increasing timestamp value. The refresh rate of the sink is selected in a way to trade-off overhead for robustness to lost interests. An interest cache is maintained at each sensor and contains the following fields:

- Timestamp of the last received matching interest.
- A gradient per neighbor indicating the data rate requested by the neighbor (1 event per second for the above interest) and a direction in which to send events.
- Approximate lifetime of the interest (10 s for the above interest).

These fields all together help a sensor maintain only unique and active interests in its interest cache. When a sensor receives an interest, it may resend it to some subset of its neighbors or even rebroadcast it to all of its neighbors in case it does not have information about the sensors that satisfy the interest. This allows interests to be *diffused* throughout the network. For energy savings purposes, geographic routing can be used in order to limit the topological scope for interest diffusion. Also, a sensor can use cached data to direct interests to particular sensors instead of broadcasting them to all of its neighbors.

A response (or event description) to the above interest is also named and could have the following form:

Type = four-legged animal // type of animal seen
Instance = elephant // instance of this type
Location = [145,222] // sensor location
Intensity = 0.6 // signal amplitude measure
Confidence = 0.85 // confidence in the match
Timestamp = 01:20:40 // event generation time

A sensor will consult its interest cache to decide to which neighbors it has to unicast this event description (or *data* message) using the highest data rate over all its outgoing gradients. Upon receiving a data message, a sensor may drop it if it does not find a matching interest in its cache. Otherwise, it checks the *data cache*, which contains the data messages seen recently, corresponding to the matching interest entry. As a result, this data message can be cached in the data cache and resent to the sensor's neighbors, if it is not already in the data cache. Otherwise, this data message is simply dropped. If the data rate specified in all gradients is higher than that of incoming events, the receiving sensor will simply send the received data message to the corresponding neighbors. Otherwise, the receiving sensor may *downconvert* to the specified data rate for the neighbors with a lower requested data rate.

At the beginning of the directed diffusion process, the sink specifies a low data rate for incoming events. After that, the sink can *reinforce* one particular sensor to send events with a higher data rate by resending the original interest message with a smaller interval. Likewise, if a neighboring sensor receives this interest message and finds that the sender's interest has a higher data rate than before, and this data rate is higher than that of any existing gradient, it will *reinforce* one or more of its neighbors. Note that it may happen that the sink reinforces one neighboring sensor A, but then receives a new event from neighboring sensor B that sends the event before A does. Directed diffusion allows the sink to *negatively reinforce* the path through A by explicitly degrading the path through A by resending the interest message with a lower data rate. Upon receiving this interest message, sensor A degrades its gradient toward the sink. Additionally, sensor A will negatively reinforce its neighbors sending at a data rate higher than those of all its gradients.

4.4.3.3 Rumor Routing. In data-centric networks, it is necessary to design efficient protocols for routing queries to the sensors that have detected the events of interest. A query can be either a request for obtaining relevant data or an order to collect data. When a coordinate system is not available or the phenomenon of interest is not geographically correlated, geographic routing cannot apply. In this case, the use of a flooding technique can be a justifiable alternative, which depends on the ratio of the number of events to the number of queries. In fact, query flooding is a useful scheme when the number of events is very high compared to the number of queries. Otherwise, event flooding is an efficient scheme. In this context, a data-centric routing protocol, called *rumor routing*, can be used as a logical compromise between query flooding and event flooding schemes [25]. Rumor routing is an efficient protocol if the number of queries is between the two intersection points of the curve of rumor routing with those of query flooding and event flooding.

Rumor routing is based on the concept of *agent*, which is a long-lived packet that traverses a network and informs each sensor it encounters about the events that it has learned during its network traverse. It seems more reasonable to only allow the sensors that have observed events to create agents that will carry relevant information to the other sensors in the network. An agent will travel the network for a certain number of hops and then die. Each sensor, including the agent, maintains an event list that has event-distance pairs, where every entry in the list contains the event and the actual distance in the number of hops to that event from the currently visited sensor. Therefore, when the agent encounters a sensor on its path, it synchronizes its event list with that of the sensor it has encountered. Also, the sensors that hear the agent update their event lists according to that of the agent in order to maintain the shortest paths to the events that occur in the network. A sensor can also send a query for a particular event. This query can be either sent along a route to the sensor that witnessed the event or forwarded in a random direction in the network.

A query stays in the network as long as its time-to-live has not expired or has not reached the sensor that has observed the target event. The agent maintains a list of recently seen sensors. Hence, when it visits a sensor, it adds the sensor's neighbors to the list and picks its next hop that is not already in the list. This will maximize the chance of creating fairly straight dissemination paths that yield better results than random forwarding. Similarly, when a query is sent in a random direction, it acts exactly in the same way as an agent. In case that the query originator finds out that the query has not reached its destination, it simply gives up or floods it in the network to guarantee its delivery.

4.4.3.4 The Cougar Approach. A design approach that is based on preprogrammed sensors and a central entity at which data is aggregated and stored for future querying and analysis suffers from two major problems. First, it is impossible for a user to change the behavior of the system on the fly, for example, to change the sensing task of the sensors. Second, the main goal of battery power conservation in the design of a network cannot be fully met when the sensed data

communicated by the source sensors to the sink is huge and correlated. In this case, the network does not benefit from in-network processing that can reduce the amount of information to be fed in the network, thus saving significant energy. To address these problems, a database approach to tasking sensor networks, called *Cougar*, was proposed in Ref. [26]. The Cougar approach provides a user and application programs with declarative queries of the sensed data generated by the source sensors. These queries are suitable for WSNs in that they abstract the user from knowing the execution plan of its queries. In other words, the user does not know which sensors are contacted, how sensed data are processed to compute the queries, and how final results are sent to the user. The Cougar approach uses a *query layer* where every sensor is associated with a *query proxy* that lies between the network layer and application layer of the sensor. This query proxy provides higher level services through queries that can be issued from a gateway node. Furthermore, the Cougar approach employs in-network processing to reduce the total energy consumption and enhance the network lifetime. Specifically, the query proxy is responsible for in-network processing, for example, aggregating records or eliminating irrelevant records, when it processes user queries. For some sensing applications, the sensed data of individual sensors are either not important or inaccurate. Hence, it is more beneficial if a set of sensed data could be aggregated or fused into a single one that is more representative and thus significant to the user.

There are a few challenges facing any database approach, including the Cougar approach. First, a network can be viewed as a huge distributed database system, where every sensor possesses a subset of data. Hence, current distributed management approaches cannot be applied directly, but need to be modified accordingly. In particular, data in a WSN could change very frequently depending on the frequency of occurrence of new events. Therefore, if a sensing application cares about the current state of the network, sensed data have to be updated frequently. One way to keep query results up-to-date is to have long-running queries that recompute query results periodically. However, all sensed data do not have the same change rate. For example, in an object detection application, data values change rapidly and thus become outdated quickly. In a temperature sensing application, data values change slowly over time. Thus, the queries issued to the sensors in these two applications do not need to have the same execution rate. For the temperature sensing application, which requires only approximate results, previous values have to be cached and the query update rate has to be lowered in order to save energy. Moreover, *uncertainty* is an inherent property of data measurement. Actually, the sensed data, for example, temperature, generated by a sensor, is an approximate as there are always errors introduced by the sensor. Another source of errors in the sensed data is noise. If we assume that this error follows some distribution, we can compute the probability that the actual value of temperature T lies in the range $[T_1, T_2]$. Hence, the user should be provided the possibility to query the network about all temperatures whose actual values lie in the range $[T_1, T_2]$ with a given probability.

Given that local computation is much cheaper than communication, the Cougar approach is in favor of in-network processing. In other words, adding computation to a sensor and reducing communication in the network will help save a significant amount of energy. In addition to the query proxy at each sensor, there is a query optimizer in the gateway, which generates distributed query processing plans once it receives queries from the outside. A query plan is disseminated to the relevant sensors after an exact computation plan at each sensor and the data flows between those sensors are specified. The processing of a query starts once the sensor behaviors are synchronized. A query plan can also specify a leader election algorithm of this query. For the temperature monitoring application, a query could be *Notify an administrator if the temperature in the office is greater than a user-defined threshold*. In this case, the query optimizer will produce a query plan for each of the relevant sensors, indicating how to elect a leader that will compute the average temperature. These query plans are disseminated to the query proxies of the relevant sensors to control their execution and the submission of their results toward the leader sensor. The leader can be elected randomly among all the relevant sensors. However, for an energy-efficiency purpose, this leader can be the sensor with the highest residual energy. It can also be energy efficient to select a leader based on its physical location. The leader selection should consider the cost of data communication from the source sensors to the leader and the data delivery cost from the leader to the gateway. Note that both costs depend on the position of the leader. There are also two computation plans, one for the leader and the other for each of the nonleader sensors. Each of the nonleader sensors has a sensor scan that reads sensor values periodically and sends them to an in-network aggregator. The aggregator combines its local data with the partially aggregated data received from other sensors and sends the results toward the leader. The query plan for the leader is to compute the average of the partially aggregated results and compare it to the user-defined threshold. If the computed average is greater than the threshold, the leader will send the average temperature value to the gateway node.

4.4.3.5 Active Query Forwarding. Another data-centric querying mechanism, called *active query forwarding in sensor networks* (ACQUIRE), was proposed for querying named data [27]. ACQUIRE is a data-centric routing mechanism for providing superior query optimization to answer specific types of queries, called *one-shot complex queries for replicated data*. In ACQUIRE, a query (i.e., interest for named data) consists of several subqueries for which several simple responses are provided by several relevant sensors. Each subquery is answered based on the currently stored data at its relevant sensor.

ACQUIRE allows a sensor to inject an active query in a network following either a random or a specified trajectory until the query gets answered by some sensors on the path using a localized update mechanism. When a sensor receives a query, it triggers an on-demand update to obtain information from all neighbors located within a look-ahead of d hops. This query is resolved incrementally as it traverses the network from one sensor to another. When it is completely

solved, it returns a completed response to the query originator. The value of d has an impact on the trade-off between the collected information, which helps reduce the length of the overall trajectory of the query, and the cost introduced by collecting the information. ACQUIRE is a mechanism for extracting data from relevant sensors in order to respond to complex, one-shot, and nonaggregate queries for replicated data. It differs from traditional flooding-based query techniques. In those techniques, several copies of a query are flooded into the network by either a querying sensor or the sink (or simply *querier*). Any sensor with relevant data will respond to this query. Note that for a one-shot query whose answer is a single value, flooding will be very costly and dominates the querying costs. Actually, the energy costs will be higher even when aggregation is used due to duplicate responses. Moreover, the querying and answering stages are separated. Unlike such query techniques, ACQUIRE allows the querier to inject a complex query into the network to be forwarded stepwise through a sequence of sensors. A sensor receiving this active query, known as *active sensor*, will partially answer it based on its local knowledge. This knowledge can be either its fresh updates/data or the updates received from all sensors within a look-ahead of d hops as a result of a request originated from the active sensor to all sensors within d hops if its data is obsolete. This request is forwarded hop by hop and each sensor receiving the request will forward its information to the active sensor. After that, the active sensor chooses its next sensor from those d hops to which it forwards the partially resolved query (or remaining query). The next active sensor selection can be randomly done by executing a random walk or by finding an appropriate sensor that would guarantee the maximum possible further resolution of the query. Following the same process, the query becomes smaller and smaller as it is forwarded from one active sensor to another until its complete resolution; that is, becomes a *completed response*. Then, the completed response is sent back to the original querier using the reverse path or the shortest path.

4.4.3.6 Energy-Aware Data-Centric Routing. One way to save the energy of sensors is to turn their radios off from time to time. However, this cannot be done to all sensors in the network. Otherwise, a sensor cannot be used as a relay to forward data on behalf of other sensors. *Energy-aware data-centric routing* (EAD) is a novel distributed routing protocol, which builds a virtual backbone composed of active sensors that are responsible for in-network data processing and traffic relaying [28]. In this protocol, a network is represented by a broadcast tree spanning all sensors in the network and rooted at the gateway, in which all leaf nodes' radios are turned off while all other nodes correspond to active sensors forming the backbone and thus their radios are turned on. Specifically, EAD attempts to construct a broadcast tree that approximates an optimal spanning tree with a maximum number of leaves, thus reducing the size of the backbone formed by active sensors. This approach is energy aware and helps extend the network lifetime. The gateway plays the role of a *data sink* or *event sink*, whereas each sensor acts as a *data source* or *event source*.

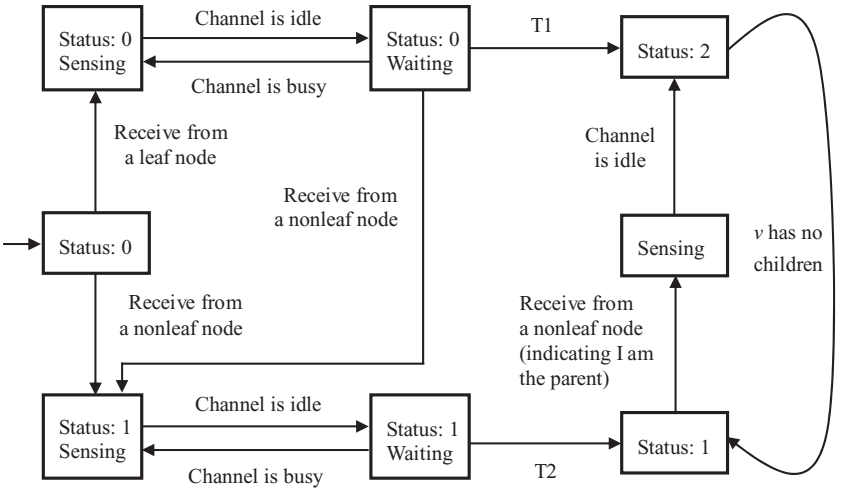

Fig. 4.8 State transition of an EAD sensor.

EAD enables the sensors and the sink to exchange messages with four fields. The state diagram of EAD is given in Fig. 4.8. If s is the sender of a message, the fields are *type* of s (sensor or sink) indicating the status of s, which can be *undefined, nonleaf node*, or *leaf node*; the *level* of s, which indicates the number of hops from s to the sink; the *parent* of s, which indicates the next hop of s in the path to the sink; and the *power* of s, which indicates the residual energy of s. A sensor switches to a specific state based on the messages it receives from other sensors. Initially, the states of all sensors are *undefined*. The sink starts by broadcasting a message $msg(2, 0, NULL, \infty)$ with ∞ indicating the sink with infinite energy. When a sensor v receives a message $msg(2, level_u, parent_u, E_u)$ from a sensor u, it becomes a leaf node, senses the channel until it is idle, and waits for time T_2^v. Then, node v broadcasts $msg(1, level_u + 1, u, E_v)$ if the channel is still idle. If sensor v receives $msg(1, level_u, parent_u, E_u)$ from sensor u, it senses the channel until it is idle, and waits for time T_1^v. Then, sensor v becomes a nonleaf node by broadcasting $msg(2, level_u + 1, u, E_v)$ if the channel is still idle. If a nonleaf node v receives $msg(2, level_w, v, E_w)$ from node w, in which it indicates that node v is its parent, after the channel is idle, node v becomes a nonleaf node by broadcasting $msg(2, level_v, parent_v, E_v)$. This message exchange takes place until the status of each sensor is either a *leaf node* or a *nonleaf node*. Obviously, any sensor with *undefined* status becomes a leaf node if it finds out that it has no children from the messages it has received.

EAD is based on the residual energy of the sensors, and provides a neighboring broadcast scheduling by T_1 and T_2, and a distributed competition among neighboring sensors to become nonleaf nodes of the broadcast tree by T_1. When a sensor v switches to a nonleaf node by broadcasting its corresponding message into the network, all of its 1-hop neighbors whose status is *undefined* become a *leaf node* and broadcast their status, and sensors with higher residual energy

broadcast first. The 2-hop neighbors of sensor v hear those broadcasts and start to compete with each other to become a nonleaf node, where the sensors with the highest residual energy win the competition. The waiting times T_1^v and T_2^v can be computed as follows:

$$T_1^v = 2t_0 + c/E_v,$$

$$T_2^v = t_0 + c/E_v,$$

where t_0 is an upper bound on the propagation time between a pair of neighboring sensors and c is an adjusting constant. This process will force neighboring sensors with higher residual energy to broadcast before those with lower residual energy.

Note that the broadcast tree grows from the sink. The identified leaf nodes of the tree will turn their radios off. However, they turn them on periodically or when they detect events. The nonleaf nodes constitute the virtual backbone that will be used by the sink to broadcast its queries to the sensors forming the backbones, which will reply by sending their data to the sink using the sensors in the backbone as relays.

A round in EAD is composed of two phases: the virtual backbone formation and data transmission to the sink. While the nonleaf nodes act as relay nodes in any communication between the sink and the source sensors, the leaf nodes are not active. Therefore, those nonleaf nodes deplete their battery power faster than the leaf nodes. Hence, when those nonleaf nodes die, their children have no parents and the broadcast tree is disconnected. Those leaf nodes will have to transmit directly to the sink, which is costly in terms of energy consumption. However, once a current round is done, the next one will start and take care of those leaf nodes, where a new virtual backbone is formed. Thus, the broadcast tree is maintained at the beginning of each round. Furthermore, the virtual backbone formation phase takes much time as messages are broadcast from the sink to all sensors in the network. To make EAD more scalable when the network size increases, a topology-based approach was proposed. In this approach, all sensors' radios are initially off. For every T_0 time, a sensor u randomly wakes up and broadcasts a hello message. An active sensor v that hears this message replies with a message containing a binary $INIT = 1$ if the number of neighbors of v is < 4; otherwise, $INIT = 0$. If u receives $INIT = 1$ or finds out that v has < 4 neighbors, it stays active; otherwise, it switches to the sleep mode. After T_0 time, EAD attempts to build a broadcast tree rooted at the sink. However, it is not guaranteed that the broadcast tree spans all active nodes. Note that the sleeping sensors wake up periodically to compute their parents. These sensors can join the broadcast tree as nonleaf nodes with the help of active neighbors that want to connect to the tree.

4.4.3.7 Information-Directed Routing. In some sensing applications, the information content of the messages exchanged by sensors is very important.

A routing protocol can successively refine the content of the exchanged messages as in a tracking application. Taking this into account, a routing protocol cannot simply be viewed as a message-forwarding mechanism. Depending on where a query is routed, two source-initiated on-demand routing protocols were proposed in Ref. [29], both of which consider a target tracking application.

In the first protocol, a user issues a query from a peripheral sensor (or entry sensor), called *query proxy*. This query is to collect information generated by sensors about a phenomenon of interest. Note that the sending sensor (or query proxy) does not know the destination that is usually known by traditional routing protocols. Therefore, the query proxy will have to find out the high information content area. In other words, the query proxy has to determine a neighboring sensor that the query should be relayed to and may have better information. In this type of routing, each relay sensor includes its own measurement to refine the mobile target estimate. The relay process looks like routing with a gradient in the information field whose content is dynamic given that every relay sensor incorporates its contribution in the form of the knowledge it has about the mobile target. In this context, a greedy approach cannot be applied to this canonical problem of target tracking due to the fact that a greedy search does not consider the choice of any information measurement. Figure 4.9 illustrates the problem that can be caused by applying a greedy routing protocol. While the target moves on a vertical line from X to Y, the relay keeps bouncing between sensors A and B because both sensors have a higher information value about the target. The relay process cannot involve any of the other sensors, for example, E, F, or G, because of the existence of a sensor hole that the mobile target went through. Assume that the mobile target keeps oscillating between X and Y for the same network configuration (Fig. 9b). To deal with the sensor hole problem, the *greedy perimeter stateless routing* (GPSR) [16] can be used. However, the relay process in this case gets away from the target that moves into and out of the hole area. Thus, the routing path should alternate between A and B. Although GPSR is well suited for static planar graphs, it is not a good choice for the mobile target

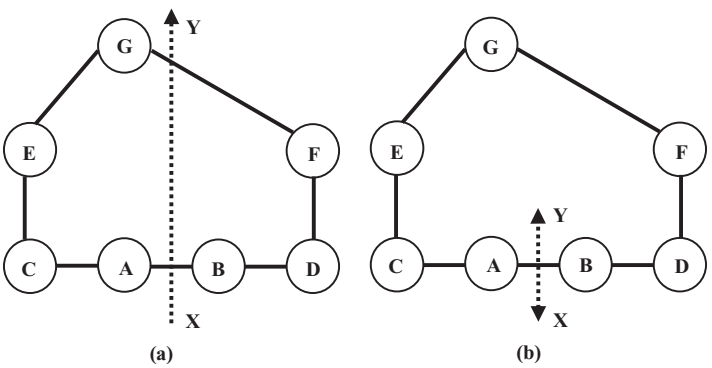

Fig. 4.9 Routing in the presence of sensor holes.

tracking scenario. To address the above problem, an information-directed multiple step look-ahead approach can be used. Specifically, a suboptimal path-finding algorithm, called *minhop* algorithm, is run by each sensor. The objective is to find a path with maximum information aggregation (or gain) among the paths with $< M$ hops. This look-ahead horizon M is selected in a way to be comparable to the diameter of the sensor hole with as low computational cost as possible. When a sensor receives a query-routing request in the form of a tuple $(t, S^{(t)}, P^{(t)})$ of time, state, and path, it wakes up and becomes active. In the case of target tracking, the state is represented by the belief $p\left(x^{(t)} \middle| \overline{z^{(t)}}\right)$, where $x^{(t)}$ is an estimate of the target location based on a set of measurements at time t denoted by $\overline{z^{(t)}} = \{z^{(0)}, z^{(1)}, \cdots, z^{(t)}\}$. Hence, the belief (or prior knowledge) that a sensor has at time t about the location of a target can be expressed by a conditional probability density function. This active sensor selects one of its M-hop neighbors with the highest information value as the destination. Given that there are several minimum hop paths from it to the selected sensor as the destination, it compares between them and chooses the path with the maximum information aggregation by maximizing the function $\Sigma_k I_k$, where I_k is the information contribution of sensor k with measurement $z_k^{(t+1)}$. I_k can be computed as follows:

$$I_k = MI\left(X^{(t+1)}; Z_k^{(t+1)} \middle| \overline{Z^{(t)}} = \overline{z^{(t)}}\right)$$

and

$$MI(U;V) = E_{p(u,v)}\left[\log \frac{p(u,v)}{p(u)\,p(v)}\right] = D(p(u|v)\|p(u)),$$

where $MI(U;V)$ is the *mutual information* [59] between two random variables U and V with a joint probability density function $p(u, v)$ and $D(\cdot\|\cdot)$ is the *Kullback–Leibler divergence* [59] between two distributions. Here mutual information is used to quantify the contribution of each sensor. Note that I_k measures the information conveyed by $z_k^{(t+1)}$ about the location $x^{(t+1)}$ of the target based on the current belief. Thus, I_k quantifies the amount of change in the posterior belief by sensor k by applying its measurement $z_k^{(t+1)}$. After that, the active sensor incorporates its measurement and sends its query to the selected destination using the selected path.

Note that the minimum hop path with the maximum accumulated information is computed after converting the graph $z_k^{(t+1)}$ representing the network. This conversion algorithm assigns to each ingoing edge a cost equal to $L - I_i$, where I_i is the information value at the incident sensor in the graph and L is a large value. Hence, the path-finding problem is turned into a shortest path problem.

In the second protocol, a query proxy will be requested through a query issued by the user to collect information from the network and report to an extraction sensor (or exit sensor). The objective is to collect as much information

as possible as the query is routed from the query proxy to the exit sensor in order to have a good estimate at the exit sensor about the state of the mobile target. Furthermore, the total communication cost should be upper bounded by a prespecified "hypothetical" cost C_0 that is considered as a *soft* constraint. This cost is used to control the trade-off between the communication cost and the information aggregation. To achieve a minimum communication cost, C_0 should be low, thus favoring the shortest path routing. Otherwise, C_0 should be high, thus leading to better information aggregation. This routing protocol is based on the basic best-fit heuristic search, denoted by A^*, in which the merit of a sensor is computed as the sum of the cost g to reach this sensor from the query sensor and the cost h (or cost-to-go) to arrive at the exit sensor. The path along which the query is routed from the query proxy to the exit sensor is guaranteed to be optimal if the estimated cost h does not exceed the true cost-to-go. For example, the Dijkstra's algorithm is a special case of A^* with estimated cost-to-go $h = 0$. For real-time path finding, there is a variation of A^*, called real-time A^* (RTA^*), which uses local information and guarantees finding a path if it exists. However, the solution may be suboptimal and the selected path may have backtracked behavior. By using RTA^*, the active sensor selects the best move based on the estimated cost-to-go h. The total cost for routing a query from the query proxy to the exit sensor is the sum of the communication cost and the information aggregation cost. The communication cost can be estimated based on the Euclidean distance metric. The information aggregation cost can be estimated based on the currently available information. Assume that we have planned the path $P^{(t)}$ and that the current node (or sensor) is v_t. In order to get to the exit sensor v_{exit}, the length of the remaining path is upper bounded by $C_0 - C_{P(t)}$ with $C_{P(t)}$ being the communication cost already paid. The region that covers all possible paths from v_t to v_{exit}, which satisfy the constraint of the communication cost, is given by an ellipse whose major and minor axes are X_1X_2 and Y_1Y_2, respectively, as shown in Fig. 4.10.

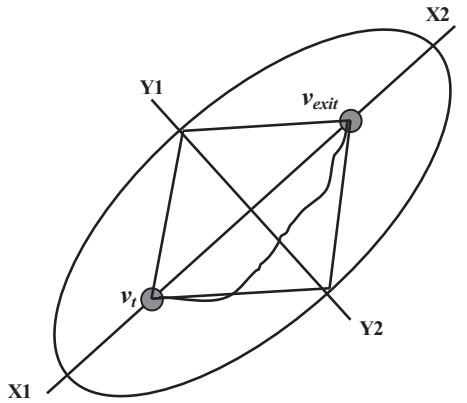

Fig. 4.10 Ellipse for all possible path coverage.

Note that there are an infinite number of paths in the region covered by the ellipse satisfying the length constraint. Thus, it would be difficult to estimate h and only four representative paths are sampled:

- Path 1: $v_t \rightarrow X_1 \rightarrow v_{\text{exit}}$
- Path 2: $v_t \rightarrow X_2 \rightarrow v_{\text{exit}}$
- Path 3: $v_t \rightarrow Y_1 \rightarrow v_{\text{exit}}$
- Path 4: $v_t \rightarrow Y_2 \rightarrow v_{\text{exit}}$

For routing the query from node v_t to the exit node v_{exit}, the information lying ahead is estimated as the maximum among the four paths. This measurement helps approximate the admissible heuristic estimate. In the case that $C_0 - C_{P(t)}$ is less than the Euclidean distance $|v_t - v_{\text{exit}}|$, the estimation of the information is equal to zero and the forward search is equivalent to a shortest path problem using the communication cost only. Likewise, the query from the query proxy to the exit sensor follows the shortest path if $C_0 = 0$.

4.4.3.8 Quorum-Based Information Dissemination. The *quorum*-based protocol [30] uses the same ADV, REQ, and DATA messages as the family of SPIN protocols. The ADV message is used to advertise some new data, the REQ message contains a request for some data, and the DATA message contains the data that were requested by a sensor. This protocol allows a sensor that wants to advertise its data to send an ADV message in both the north and south directions so that the data can reach both the north and south boundaries of the network. On the other hand, a sensor that wants to access the data sends its REQ message into both east and west directions. Note that an ADV message is sent out to a group of sensors while a REQ message is sent out to another group of sensors. Assuming that there exists a sensor that belongs to both groups, this sensor is called a *rendezvous* node that has received both the ADV and REQ messages. Recall that the ADV message contains the location of the sending sensor and the data description, whereas the REQ message contains the location of the querying sensor and the description of its interest. Thus, the rendezvous sensor first checks whether there is a match between the interest in the REQ message and the description in the ADV message. If there is a match, the rendezvous sensor sends a REQ message to the source sensor that initiated the ADV message along the path traversed by the matched ADV message, which in turn sends a DATA message back to the querying sensor that originated the REQ message along the path traversed by the REQ message.

When a neighbor receives an ADV message originally sent by a sensor s, it will record the ADV message along with the sending sensor s. For the northward direction, the sensor with the highest y coordinate is selected to further forward the ADV message to its neighbors. This kind of greedy geographical forwarding continues until the ADV message reaches the farthest sensor at the north boundary. Similarly, the same process will take place for the southward direction where

the ADV message propagates until it reaches the south boundary. Likewise, when a sensor is interested in an advertised data, it will broadcast a REQ message in both eastward and westward directions until it reaches the farthest sensors at the east and west boundaries, respectively. Therefore, the sensors located close to the intersection of the vertical line that passes through the source of the ADV message and the horizontal line that passes through the querying sensor (or sender of the REQ message) have already received both the ADV and REQ messages. These sensors are the rendezvous sensors for the matched pair (ADV, REQ). When one of these rendezvous sensors receives a new REQ message, it checks whether it has already received an ADV message matching that REQ message. If there is a match, the rendezvous sensor will forward the REQ message to the source of the ADV message using the reverse path of the one along which the matched ADV message was sent by its source. Then, the source sensor will send a DATA message to the querying sensor using the reverse path of the one along which that REQ message was sent. When the rendezvous sensor receives the DATA message, it will forward it to the sensor that initiated the REQ message.

Note that the rendezvous sensors are in charge of relaying REQ messages to the source of the matched ADV messages and forwarding DATA messages to the initiators of the matched REQ messages. Since the sensors are supposed to be location aware, the source sensor can use one of the existing geographical routing protocols to deliver the DATA messages directly to the querying sensor without passing by the rendezvous sensor, that is, without using the row and column routes created by the quorum system. This option will help balance the data dissemination load on all the sensors in the network.

4.4.3.9 Home Agent-Based Information Dissemination. In home agent-based information dissemination [30], all sensors in a network are associated with a particular sensor or a location, called *home agent*. Specifically, this home agent could be a real sensor or a geographical area containing a number of sensors. In the quorum-based protocol, the rendezvous sensors are associated with pairs of ADV and REQ, and hence all sensors are candidates to act as rendezvous sensors for data dissemination. In contrast, there is only one home agent for all sensors in the home agent-based protocol. Any ADV message is sent to the home agent and any REQ message is also sent to the home agent. Hence, a home agent acts as a point of contact between the source sensors (i.e., senders of ADV messages) and the querying sensors (i.e., senders of REQ messages).

The selection of a home agent is performed *a priori*. If it is a sensor, a home agent can be selected in a way to minimize the maximum distance between any sensor and the home agent. This distance can be expressed in terms of either the Euclidean distance between two sensors or the number of hops between them. When it is a location, a home agent is the center of the sensor field. Using a geographical routing protocol, sensors can send their ADV and REQ messages to the home agent that will act as the rendezvous sensor for ADV and REQ messages as described earlier. In case of choosing the center of the sensor field as a home agent, it may happen that there is no sensor located at the center of

the field. In this case, the ADV/REQ messages are propagated within a circle centered at the center of the field with some radius. One of the sensors inside the circle will be designated as the rendezvous sensor. In case the circle is empty, that is, a void region, a routing protocol, for example, GPSR [52], can be used to route the ADV/REQ messages around the void area and select a rendezvous sensor for ADV and REQ messages.

Note that the home agent cannot span the entire network as it is the case for the rendezvous sensors in the quorum-based protocol. All the sensors located around the home agent are heavily used in data dissemination, and hence deplete their battery power very quickly, resulting in the energy sink-hole problem.

4.4.4 Multipath-Based Protocols

Considering data transmission between source sensors and the sink, there are two routing paradigms: *single-path* routing and *multipath* routing. In single-path routing, each source sensor sends its data to the sink via the shortest path. In multipath routing, each source sensor finds the first k shortest paths to the sink and divides its load evenly among these paths. In this section, we review a sample of multipath routing protocols for WSNs.

4.4.4.1 Disjoint Paths. First, we consider the design of multipath protocols that help find a small number of alternate paths that have no sensor in common with each other and with the primary path. These protocols are said to be *sensor-disjoint* multipath routing [31,32]. In sensor-disjoint path routing, the primary path is best available whereas the alternate paths are less desirable as they have longer latency. Being disjoint makes those alternate paths independent of the primary path. Thus, if a failure occurs on the primary path, it remains local and does not affect any of those alternate paths. In general, multipath routing leads to the construction of k node-disjoint multipaths by assuming a global knowledge of the network topology. However, these k disjoint paths can also be constructed in a localized manner. Actually, the sink can determine which of its neighbors can provide it with the highest quality data characterized by the lowest loss or lowest delay after the network has been flooded with some low-rate samples. Then, the sink sends out a *primary-path reinforcement* to its best neighbor. This neighbor can apply the same mechanism as in the case of directed diffusion to locally identify its most preferred neighbor. This reinforcement process repeats until the construction of the primary path is done. After that, the sink iterates the same operation for its next most preferred neighbor by sending out to it an *alternate-path reinforcement*. If each sensor accepts only the first reinforcement, those alternate paths are guaranteed to be disjoint with each other and with the primary path. In other words, a sensor negatively reinforces all reinforcements that follow the first one. Note that there is no guarantee that this search procedure of *localized* disjoint paths will discover the same alternate paths as in the idealized version that assumes a global knowledge of the network topology.

4.4.4.2 Braided Paths. Although disjoint paths are more resilient to sensor failures, they can be potentially longer than the primary path and thus less energy efficient. Furthermore, they introduce higher delay when they replace the primary path. Relaxing the disjointedness constraint leads to partially disjoint paths from the primary one, called *braided multipath* [31,32]. To construct the braided multipath, the first step is to compute the primary path. Then, for each node (or sensor) on the primary path, the best path from a source sensor to the sink that does not include that node is computed. As can be seen, those best alternate paths are not necessarily disjoint from the primary path. This set of alternate and primary paths are called *idealized* braided multipaths. Moreover, the links of each of the alternate paths lie either on or geographically close to the primary path. Therefore, the energy consumption on the primary and alternate paths seems to be comparable as opposed to the scenario of mutually disjoint alternate and primary paths.

Similarly, the braided multipath can be constructed in a localized manner. First, the sink sends out a primary-path reinforcement to its first preferred neighbor and an alternate-path reinforcement to its second preferred neighbor. Also, each of the other nodes on the primary path sends out an alternate-path reinforcement to its next most preferred neighbor. Hence, each node on the primary path attempts to route around its immediate neighbor on the primary path toward the source. Furthermore, when a node receives an alternate-path reinforcement, it drops it or propagates it further to its most preferred neighbor depending on whether that node lies on the primary path or not.

4.4.4.3 N-to-1 Multipath Discovery. For the disjoint and braided multipaths previously discussed, the objective is to find multiple disjoint or partially disjoint paths between a source sensor and the sink. This is the case for most of the multipath routing protocols. In Ref. [33], an *N-to-1 multipath discovery* protocol is proposed, which benefits from flooding to find multiple node-disjoint paths from each sensor to the sink simultaneously. This protocol is based on the simple flooding originated from the sink and is composed of two phases, namely, *branch aware flooding* (or phase 1) and *multipath extension of flooding* (or phase 2). Both phases use the same routing messages whose format is given by {*mtype*, *mid*, *nid*, *bid*, *cst*, *path*}, where *mtype* refers to the type of a message. During phase 1, all paths are primary, which is indicated by *mtype* = *"RPRI"; mid* represents a sequence number of the current routing update; *nid* is the ID of the sender of the message; *bid* is the ID of the branch defined as the ID of the closest node (*nid*) to the sink in the branch; *path* is a sequence of nodes visited by the message; and *cst* is the cost of the path. A message {*RPRI*, *mid*, *Sink*, Ø, 0, (*Sink*)} is broadcast by the sink periodically or on-demand in order to initialize the routing update. Upon hearing a message {*RPRI*, *mid*, *nid*, *bid*, *cst*, *path*} for the first time, a sensor *s* determines its parent (*nid*) and rebroadcasts an updated version of the received message in the form of {*RPRI*, *mid*, *s*, (*bid* = Ø)?*s* : *bid*, *cst* + cost (*s*, *nid*), *path* + (*s*)}. Furthermore, the sensor *s* marks the path *p* = *path* + (*s*) as the *primary path* back to the sink. Note that the sensor *s* updates the field *bid* only if it is

empty. Also, the cost is incremented by the cost from the sensor s to its parent (*nid*) and the path includes the receiving sensor s. When the sensor s receives the same message from a neighbor s', it marks s' as a *child* or *sibling* based on the content of the path if $bid = s$; otherwise, the message is coming from another branch. Thus, s marks s' as a *cousin*. When the sensor s receives a message from its cousin, it checks if the path $q = path + (s)$ is disjoint with the primary path p and with any other alternate path with a lower cost in the alternate path set, denoted by Q_s. If this is true, the new path q is added to Q_s. In addition, any path that shares some nodes with q and has a higher cost than q will be removed from Q_s. The forwarding process of RPRI messages terminates when each sensor has broadcast the message only once.

While the maximum number of node-disjoint paths from any node to the sink is upper bounded by the number of branches in the spanning tree created by flooding, the links established between the nodes belonging to different branches lead to alternate disjoint paths to the underlying nodes (i.e., the nodes being connected). Figure 4.11 gives an example showing that node w has one primary path (w, r, l, g, d, *Sink*) and an alternate path disjoint with the primary path after it has heard the broadcast by node v, thus creating a link between v and w. Similarly, the node v will also establish an alternate path to the sink through node w. Phase 2 of the protocol allows the nodes to exchange the same message format, but with the type field set to *mtype* = "*RALT*" given that the paths formed in this phase are the alternate paths. These *RALT* messages will allow a node to further propagate the alternate paths to its parent and sibling or cousin node. A sensor s composes a *RALT* message {*RALT*, *mid*, s, q, *bid*, *q.cst*, q} for each alternate path q and broadcasts it to its neighbors. When a sensor s receives a message {*RALT*, *mid*, *nid*, *bid*, *cst*, *path*}, it will learn an alternate path $q = path + (s)$ only if this message was not sent from its parent and s is not included in the path. Also, this path will be added to the alternate path set Q_s of s if q is disjoint with all the paths with a lower cost in Q_s. If this is true, sensor s

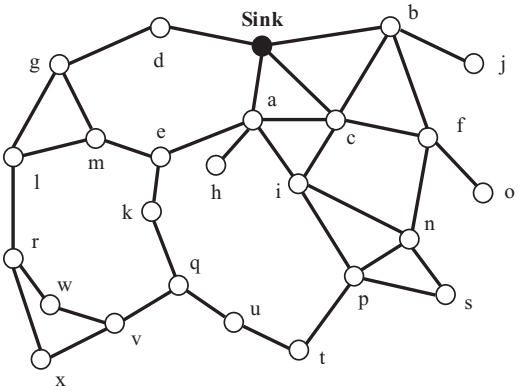

Fig. 4.11 Multipath extension of flooding.

will remove all paths in Q_s that have a higher cost and intersect with q and broadcast an updated version of q:{*RALT*, *mid*, *s*, *q.bid*, *q.cst*, *q*}. The forwarding of *RALT* messages terminates when no new disjoint alternate path can be added to the alternate path set of any node. For example, if node *w* further broadcasts the disjoint paths it learned to its neighbors, node *r* will learn a new alternate path (*r*, *w*, *v*, *q*, *k*, *e*, *a*, *Sink*) to the sink. Therefore, phase 2 helps the sensors discover more disjoint alternate paths at the cost of additional routing messages with regard to the alternate paths found during phase 1 across multiple branches.

This multipath discovery protocol generates multiple node-disjoint paths for every sensor. In multihop routing, an active per-hop packet salvaging strategy can be adopted to handle sensor failures and enhance network reliability. Assume that a network uses a reliable MAC protocol, for example, IEEE 802.11, which acknowledges the successful transmission of each frame. This would help the sensors decide whether they need to keep or remove the most recently transmitted frames from their transmission buffers. When a source sensor sends a packet toward the sink, it includes the source routing option. If at any intermediate sensor transmitting to the next hop is not successful, the current sensor will salvage the packet by randomly selecting a route to the sink and sending this packet along this route. If all next hops from the current sensor fail, the packet should be dropped. However, a routing loop can occur if the new randomly selected path consists of a sensor that the packet has already visited. In order to avoid this problem, the current node has to make sure that there is no common node between the partial route the packet has already traveled and the new candidate route used to salvage the packet. Moreover, the current node needs to update the source routing option so that the packet has the actual path it has gone through when it reaches the sink. Thus, the per-hop salvaging using alternate paths is an effective and efficient technique to handle sensor failures and enhance network reliability. Figure 4.12 illustrates a packet originally sent by sensor *t* being salvaged at sensors *p* and *f* before it reaches the sink.

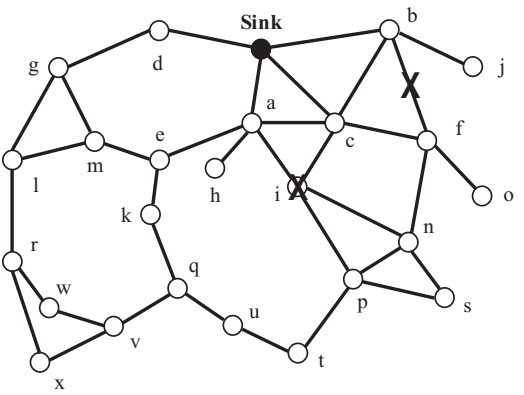

Fig. 4.12 Alternate path packet salvaging.

4.4.5 Mobility-Based Protocols

Mobility brings new challenges to routing and data dissemination in WSNs. In particular, sink mobility requires energy-efficient protocols to guarantee data delivery originated from source sensors toward mobile sinks. This section discusses several routing and data dissemination protocols for mobile WSNs.

4.4.5.1 Joint Mobility and Routing Protocol. A network with a static sink suffers from a severe problem, called *energy sink-hole problem*, where the sensors located around the static sink are heavily used for forwarding data to the sink on behalf of other sensors. As a result, those heavily loaded sensors close to the sink deplete their battery power more quickly, thus disconnecting the network. This problem exists even when the static sink is located at its optimum position corresponding to the center of the sensor field [34]. To address this problem, a mobile sink for gathering sensed data from source sensors was suggested [34]. In this case, the sensors surrounding the sink change over time, giving the chance to all sensors in the network to act as data relays to the mobile sink and thus balancing the load of data routing on all the sensors. Under the shortest-path routing strategy, the average load of data routing is reduced when the trajectories of the sink mobility correspond to concentric circles (assuming that the sensor field is a circle). Another category of mobility trajectories is to move the sink in annuli. However, such movement can be viewed as a weighted average over the movements on a set of concentric circles. In particular, the optimum mobility strategy of the sink is a symmetric strategy in which the trajectory of the sink is the periphery of the network. This result was shown by comparing mobility trajectories on concentric circles of different radii and it was proved that the maximum average load of data routing is achieved at the network center. Therefore, the trajectory with a radius equal to the radius of the sensor field maximizes the distance from the sink to the center of the network that represents the hot spot.

Another dimension of the design strategy that reduces the network load is routing. Regardless of the shape of a sensor field, it is always true that the sensors located on the network periphery are almost not used for forwarding sensed data to the sink on behalf of other sensors and thus have a longer lifetime than all other sensors in the network. Therefore, an efficient routing strategy should exploit the available energy of those sensors close to the border of the network in order to balance the data dissemination load on all the sensors. Based on the sink mobility strategy and the routing strategy discussed earlier, an energy-efficient heuristic for joint routing and mobility is to have the sink moving on a circle of radius $R_m < R$, where R stands for the radius of the sensor field, while data routing depends on the location of the source sensors. Note that the sensor field is divided into two regions: the inner circle and the annulus between the periphery of the network and the trajectory of the sink represented by a circle. The sensors within the inner circle use the shortest path routing to transmit their sensed data to the sink, whereas the sensors in the annulus send their data to the sink using two steps. First, a sensor uses *round routing* around the center of the

network O until the segment OB is reached, where B is the current position of the mobile sink. Then, the data is sent to the sink using a shortest path. This joint heuristic leads to lower network load by reducing the distance between the mobile sink and the sensors that follow the shortest path routing from R, which corresponds to the optimum mobility strategy, to R_m.

4.4.5.2 Data MULES Based Protocol. Although sensor deployment depends on the sensing application, most of the routing and data dissemination protocols for WSNs assume that sensors are very densely deployed in a network. In fact, one of the most desirable features of sensor deployment is network connectivity by which any source sensor is able to communicate directly or indirectly with the sink in order to report its sensed data. To guarantee network connectivity, two different deployment approaches can be used. First, a network can be deployed by using a large number of sensors, yielding a densely connected network in which the source sensors send their sensed data to the sink through multihop communication paths including other intermediate sensors. Second, a network can be deployed by using multiple sinks that cover the entire geographical area, where the source sensors communicate directly with the nearest sink to report their sensed data. While the first approach may not be cost-effective to build a dense and fully connected network, the second one is not cost-effective either, but reduces the communication cost that would be incurred when the sensors communicate with only one single sink. Both scenarios raise the need for developing an architecture that benefits from the two approaches and can guarantee cost-effective connectivity in a sparse network while reducing the energy consumption of the sensors.

To address this need, a three-tier architecture based on mobile entities, called *mobile ubiquitous LAN extensions* (MULEs), was proposed to collect sensed data from source sensors in sparse networks [35]. The MULEs architecture has three main components. The bottom layer contains static wireless sensors that are responsible for sensing an environment; the top layer includes WAN connected devices and access points/central repositories for analyzing the sensed data. Moreover, these access points can be positioned at locations providing network connectivity and power. They communicate with a central data warehouse enabling them to synchronize the collected data, identify redundant data, and acknowledge the receipt of the data sent by the MULEs for reliable data transmission. The middle layer has mobile entities (MULEs) that move in the sensor field and collect sensed data from the source sensors when in proximity to deliver them to those access points when in close range. These MULEs have the capabilities to communicate with both the access points and the sensors using short-range wireless communications. Hence, the MULE component can be considered as a mobile transport agent that connects heterogeneous nodes, namely, the source sensors and the access points. Furthermore, the MULEs can communicate with each other, thus forming a multihop MULE network that can be used to reduce the latency between MULEs and those access points. Because of their motion, the MULEs are able to collect and store data from the sensors, and

acknowledge them. This implies that the MULEs are equipped with larger storage capacities compared to the sensors.

Note that the MULE architecture helps the sensors save their energy as much as possible and thus extend their lifetime. Since the sensors directly communicate with the MULEs through short-range paths, they deplete their energy slowly and uniformly. Thus, the MULE architecture has low sensor energy consumption. Moreover, the MULEs move in the sensor field in a random fashion, which guarantees that all the sensors are equally visited and consume the same amount of energy during their monitoring task. In addition, the MULE architecture has low infrastructure cost. Because of the direct communication between the source sensors and the MULES, there is no routing overhead that would drain the energy of the sensors. As far as robustness and scalability are concerned, the MULE architecture is fault tolerant and scale well. If a MULE fails, it will not affect any particular sensor because no sensor is dependent on any MULE. However, it will degrade the performance of a sparse network for decreasing its data success rate and increasing its latency. Also, when the number of sensors or the number of MULEs increases, there is no need for any network reconfiguration. Note that the sensors have to wait until the MULEs come close by to report their sensed data to the MULEs. Therefore, for time-critical applications, the MULE architecture may introduce an undesirable delay in reporting the sensed data of the source sensors and thus may not be practical. One way to solve this problem is to equip the MULEs with an always-on connection so that they act as mobile sinks (i.e., MULEs and access points). Furthermore, when a MULE fails, the corresponding sensed data will never reach the sink.

4.4.5.3 Two-Tier Data Dissemination. Existing mobile sink-based routing protocols, for example, directed diffusion [36,37], require that each mobile sink propagate its location updates throughout the sensor field in order to inform all the sensors about the direction of sending future data reports. This type of information flooding increases collision in wireless transmissions and yields significant depletion of the limited battery power of the sensors. To alleviate this problem, Ye et al. [36,37] proposed a *two-tier data dissemination* (TTDD) protocol that provides scalable and efficient data delivery to multiple mobile sinks. In TTDD, while the sensors know their own locations, the sinks may or may not be aware of their own locations. Furthermore, the sensors are stationary and aware of their missions, which change infrequently. Therefore, the overhead for mission dissemination is negligible compared to that of sensed data delivery.

The TTDD protocol has three main phases: *grid construction, two-tier query, data forwarding*, and *grid maintenance*. When a source sensor has sensed data to send, it builds its own grid structure for data dissemination, in which the location of the source sensor is one of the crossing points, called *dissemination points*. Given the location of the source sensor $L_s = (x, y)$, these dissemination points are defined by the set L_p, which is given by

$$L_p = \{(x_i, y_i) | x_i = x + i\alpha;\ y_j = y + j\alpha;\ i, j = \pm 0, \pm 1, \pm 2, \ldots\}.$$

Then, the source sensor sends its data announcement message to its four adjacent crossing points that are computed by the source sensor based on its location and the size α of the grid cell. The source sensor uses greedy geographical forwarding to forward the message to the closest neighbor to the crossing points. The neighbor node will use the same forwarding technique until the data announcement message gets received by the sensors closer to the dissemination points L_p than all their neighbors. If the distance between any of these sensors and L_p is less than a threshold $\alpha / 2$, it becomes a *dissemination node* serving a dissemination point in L_p. Upon receiving a data announcement message, a dissemination node stores the source message, the dissemination point it is serving, and the location of the upstream dissemination node. Then, it further forwards the data announcement message to its neighboring dissemination points on the grid except its upstream dissemination point from which it has received the data announcement message. The announcement propagation process continues until each dissemination point is served by a dissemination node. Note that those dissemination points act as reference locations for selecting dissemination nodes. Also, the grid is built on a per-source-sensor basis, thus yielding different sets of dissemination nodes for those source sensors. This approach balances the data dissemination load among all the sensors in the network, enhances its scalability, and provides better robustness in the presence of sensor failures.

The sink can then flood its query within a cell of the grid to receive data from the source sensor. This query will eventually be received by the nearest dissemination node, called *immediate dissemination node*, from the sink. To restrict flooding of its query, the sink includes a maximum distance beyond which any sensor receiving the query will just drop it. The immediate dissemination node will forward the query to its upstream dissemination node from which it has received the data announcement message. This process repeats until the source sensor receives the query. The two-tier query forwarding process has two levels of aggregation. When an immediate dissemination node receives multiple queries from different sinks for the same data (i.e., source sensor), it sends only one query to its upstream dissemination node in the form of an *upstream update*. Likewise, when a dissemination node receives multiple upstream updates from different downstream neighbors, it further forwards only one of them. As can be seen, this two-level aggregation provides scalability with the number of sinks. A dissemination node keeps sending upstream update messages periodically in order to receive data continuously until the sink either stops sending queries or moves out of the local region. As an upstream update message traverses the grid, the dissemination nodes store soft-state timers to forward data streams back to the sinks in the reverse path. These soft-state timers are an order-of-magnitude higher than the time interval between data messages. The rationale behind this design choice is to balance the overhead caused by the forwarding of periodic upstream update messages and that introduced by sending data to the nodes where they are not useful anymore.

When a source sensor receives upstream updates from its neighbor dissemination nodes, it sends back the data to each of those dissemination nodes. These

dissemination nodes forward the data toward the nodes from which they received the upstream updates. This forwarding process continues until the data reach the immediate dissemination node of each sink. Furthermore, if a dissemination node has aggregated queries, it will send a copy of the data to each of its downstream dissemination nodes that sent those queries. In case of relaying data to a mobile sink, a forwarding technique, called *trajectory forwarding*, is used by an immediate dissemination node. With trajectory forwarding, a sink is associated with a *primary agent* and an *immediate agent*. First, a sink chooses one sensor as its primary agent and uses the location of this node in its queries. Initially, both primary and immediate agents are the same sensor. When a sink decides to move out of the range of its current immediate agent, it selects another neighboring sensor as its immediate agent by broadcasting a solicit message and sends the location of this node to its primary agent. This selection is based on the strength of the signal-to-noise ratio of the replying sensors. Any future data will be forwarded to the new immediate agent. The sink selects its immediate agent. It may happen that when a sink is about to move, data were already forwarded to its old immediate agent. To receive these data, the sink also sends the location of its new immediate agent to its old one. Therefore, the immediate dissemination node for the sink forwards the data to the primary agent of the sink, which in turn forwards them to the immediate agent of the sink. This agent is one-hop away from the sink, and hence relays the data directly to the sink. When the sink moves away from its current primary agent (e.g., the location of the new sink is one cell size from the primary agent), it selects a new primary agent by flooding a query locally. Similarly, a sink associates its primary agent with a timer that is set to the duration the sink can stay in a cell. This mechanism avoids receiving duplicate data from its old primary agent. Also, the old immediate agent is associated with a shorter timer whose value is equal to the duration a sink remains within the one-hop distance.

It is worth noting that a grid has a lifetime that is set up by a source sensor. If the grid lifetime expires before receiving any data announcement messages to extend the lifetime, all dissemination nodes clear the grid states. The grid lifetime depends on the mission of the network and the period of data availability. Moreover, to deal with sensor failures, TTDD employs a mechanism, called *upstream information duplication*, where each dissemination node selects several sensors from its neighborhood and replicates in them the location of its upstream dissemination node. If any failure of this dissemination node occurs, the upstream update message from its downstream dissemination node will be processed by one of those selected sensors. This selected sensor will emerge as a new dissemination node and will forward the update message to the upstream dissemination node based on the stored information.

4.4.5.4 Scalable Energy-Efficient Asynchronous Dissemination. To improve TTDD, a distributed self-organizing protocol, called *scalable energy-efficient asynchronous dissemination* (SEAD), was proposed in [52] to trade-off between minimizing the forwarding delay to a mobile sink and energy savings.

SEAD considers data dissemination in which a source sensor reports its sensed data to multiple mobile sinks and consists of three main components: dissemination tree (d-tree) construction, data dissemination, and maintaining linkages to mobile sinks. Also, SEAD does not assume high network density and does not use flooding to find an entry to the d-tree for a mobile sink joining d-tree. However, it assumes that the sensors are aware of their own geographic locations. Every source sensor builds its data dissemination tree rooted at itself and all the dissemination trees for all the source sensors are constructed separately. SEAD can be viewed as an overlay network that sits on top of a location-aware routing protocol, for example, geographical forwarding.

In SEAD, every mobile sink is associated with one of its neighbors, called *access node*, which will be responsible for sending a join request to a source of the d-tree on behalf of its mobile sink. Therefore, when a source sensor reports its sensed data, the access point receives the data and delivers them to its mobile sink. For this purpose, the access point keeps track of the current location of its mobile sink. While a mobile sink is not a member of the d-tree, it is represented by its access node. An access node is selected in a way such that the hop count to its mobile sink does not exceed a certain threshold used to trade-off between the energy consumed on reconfiguring the tree and the path delay. The sensed data of each source sensor is replicated at selected nodes, called *replicas*, which are located between the source sensor and the sinks. They are members of the d-tree. These replicas act as intermediate destinations for the sensed data. A d-tree is a minimum-cost weighted Steiner tree enabling the selection of replicas at intermediate points different from the source sensors and mobile sinks in order to reduce the cost of the d-tree. Figure 4.13 shows the elements of SEAD. All nodes in the d-tree collaborate to disseminate the sensed data to the mobile sinks along the tree in an asynchronous manner. For reliability purposes, each source sensor sends *idle* messages along the tree at a minimum update rate U_m in case it has no new sensed data to report. Moreover, every member of the d-tree has a pointer to each of its children and its parent as well. Thus, when a member of the d-tree does not receive any message within $1/U_m$ time units, it contacts its parent. In case of the parent's failure, the node sends out an error message to the root of the d-tree, requesting a new parent. This mechanism is used to track packet loss and node failures.

The design of the SEAD protocol includes four main phases. In the first phase, *subscription query*, a mobile sink B_i selects the closest neighbor A_i as an access node and issues a *join* query to a source sensor through its selected static access node. This join query message includes the $B_i's$ desired update rate and $A_i's$ location. Then, the access node A_i uses the routing protocol to send this message to the source sensors. Figure 4.13 shows the SEAD tree model.

The second phase, *gate replica search*, consists of determining a gate replica that acts as a grafting point on the data dissemination tree. This d-tree is extended with a new branch from this replica to the new access node. This replica will be connected to the new access node in order to feed it with the sensed data. Hence, this replica r is selected in a way such that it introduces the least additional cost

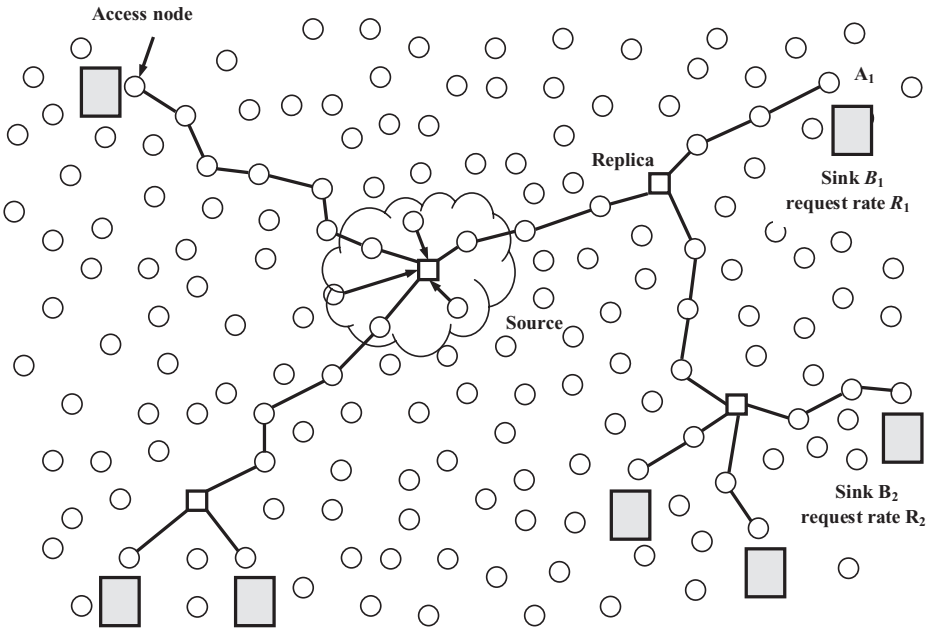

Fig. 4.13. The SEAD tree model.

$K(r)$ for connecting it to the access point. This additional cost $K(r)$ depends on the desired update rate R_i of the sink, the physical distance between the candidate replica and the access node, and the cost of the children of the candidate replica. Let E_r be a set of ancestors of r. In case the node r has a parent and the downstream rate $Q_r^{p(r)}$ of the parent $p(r)$ of r is $<R_i, Q_r^{p(r)} = R_i$. Formally, $K(r)$ is calculated as

$$K(r) = R_i d(r, A_i) + \sum_{m \in E_r} \left\| R_i - Q_m^{p(m)} \right\| d(p(m), m),$$

where

$$\|z\| = \begin{cases} z & if\ z \geq 0 \\ 0 & if\ z < 0 \end{cases}$$

and $p(m)$ is the parent of replica m. In order to calculate $K(r)$, the node r computes the incremental cost $K(r) - K(c)$ for each of its children c, that is, $r = p(c)$, as follows:

$$K(r) - K(c) = R_i d(r, A_i) - R_i d(c, A_i) - \left\| R_i - Q_c^r \right\| d(r, c),$$

where Q_c^r is the downstream rate of the child c. If $K(r)$ is $<K(c)$ for each child c in the set $C(r)$ of the children of the node r, the replica r is selected as the gate replica. Otherwise, the node r forwards the message to one of its children that maximizes $K(r) - K(c)$. It may happen that the message is recursively forwarded until it reaches a leaf node (i.e., another access node). In this case, the parent of this access node is selected as the gate replica.

The third phase, *replica placement*, is to save more communication energy by locally readjusting the dissemination tree (or d-tree) in the neighborhood of the gate replica. This will lead to an optimal d-tree from the source sensor to the access node. There are two modes for connecting the access node to the replica gate: *nonreplica mode* and *junction mode*. In the *nonreplica mode*, the access node is directly connected to the gate replica as a child. In the *junction mode*, a child for the gate replica is created and the access node is connected to the gate replica via its child replica, called junction replica, which sends sensed data to the access node as well as some of the original children of the gate replica. The selection of the appropriate mode should minimize the cost for joining the access node to the d-tree. For this purpose, the gate replica g compares between the energy cost of the nonreplica mode, that is,

$$U_{nonreplica}(c) = d(g, A_i)R_i + d(g, c)Q_c^g$$

for each child $c \in C(g)$ when the gate replica g is the parent of the access node A_i, and the energy of the junction mode, that is,

$$U_{jreplica}(c) = \min_{n \in W}\{d(g, n)\max(R_i, Q_c^g) + d(n, A_i)R_i + d(n, c)Q_c^g\},$$

where W is a set of neighbors of the gate replica g. Then, the gate replica g identifies one of its children $c \in C(g)$ that maximizes $U = U_{nonreplica}(c) - U_{jreplica}(c)$. If $U < 0$, the gate replica is directly connected to the access node. Otherwise, the child c is the sibling of the access node A_i. Then, a *jreplica_search* message that indicates node c is forwarded to the neighbor n, which in turn repeats the above process with respect to its neighbors and forwards the message recursively. This process terminates if dead ends are met, in which case a control message is sent upstream and the previous node is a junction replica. Also, if no neighbor can make $U_{jreplica}$ smaller, the current node is selected as the new junction replica J. The selected replica J stores the downstream rate Q_c^J and the desired update rate R_i of the sink. It also registers the access node A_i as its child and the gate replica g as its parent. Moreover, g sets $Q_J^g = \max\{R_i, Q_c^g\}$.

The fourth phase, *d-tree management*, is to maintain connectivity between the mobile sinks and their access nodes. The selection of access nodes depends on a threshold on the total hop count. When a mobile sink renews its access node, it sends a *disconnect* message to its old one. The old access node informs its parent in the d-tree that it has left the tree. Also, depending on the distance between the new access node and the old gate replica, the mobile sink may or may not request the source sensor for a new gate replica.

4.4.5.5 Dynamic Proxy Tree-Based Data Dissemination. The data dissemination protocols previously discussed, for example, directed diffusion and TTDD, are not efficient enough for some sensing applications, for example, mobile target detection and tracking, where the sensed data have to be disseminated from a dynamic source to multiple mobile sinks. While the directed diffusion protocol requires that source sensors flood the availability of their data throughout the entire network, thus leading to much redundancy, the TTDD protocol maintains a grid-based propagation structure over the entire network proactively irrespective of the locations of the sinks, thus causing a considerable amount of overhead. Furthermore, even the tree-based multicasting protocol is not relevant due to the frequent movement of source sensors and sinks as well as the limited communication ranges of the sensors. In the real world, paths between those sources and sinks may be disconnected frequently, which prevents much of the sensed data from reaching the sinks. Therefore, it is necessary to reconfigure the tree in order to re-stablish routes between dynamic source sensors and mobile sinks, which would cause high maintenance overhead. To address this problem, a *dynamic proxy tree-based data dissemination* framework was proposed in Ref. [39] for maintaining a tree connecting a source sensor to multiple sinks that are interested in the source. This helps the source disseminate its data directly to those mobile sinks.

In this data dissemination framework, a network is composed of stationary sensors and several mobile hosts, called *sinks*. The sensors are used to detect and continuously monitor some mobile targets, while the mobile sinks are used to collect data from specific sensors, called *sources*, which may detect the target and periodically generate detected data or aggregate detected data from a subset of sensors. Because of target mobility, a source may change and a new sensor closer to the target may become a source. Each source is represented by a *stationary source proxy* and each sink is represented by a *stationary sink proxy*. Figure 4.14 illustrates this framework. It is worth mentioning that the source and sink proxies are temporary in the sense that they change as the source sensors change and the sinks move. A source will have a new source proxy only when the distance between the source and its current proxy exceeds a certain threshold. Likewise, a sink will have a new sink proxy only when the distance between the sink and its current proxy exceeds a certain threshold. Moreover, the proxies associated with the same source sensor form a *proxy tree*. The motivation behind the design of such proxies is to reduce the cost of pushing data to and querying data from the source and sink proxies. Since the source sensors are changing and the sinks are moving, the tree should be reconfigured in order to minimize the cost of data multicasting from a source proxy to the sink proxies. For this purpose, both centralized and distributed tree reconfiguration algorithms have been proposed with an objective to minimize the cost of data dissemination and the overhead of tree reconfiguration [39].

Generating a minimum-cost proxy tree is equivalent to constructing a minimum Steiner tree connecting terminals in a graph. In this context, there are two centralized algorithms for reconfiguring a proxy tree: off-line and on-line. The

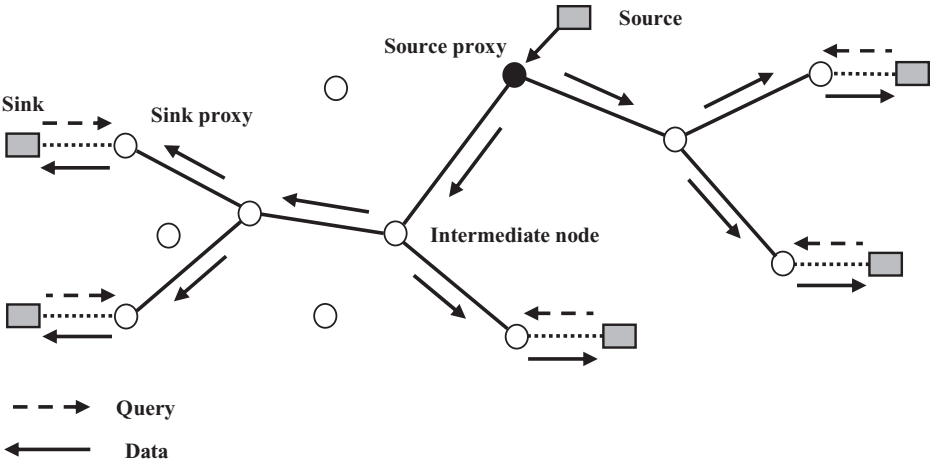

Fig. 4.14 Proxy tree for supporting dynamic multicasting.

off-line algorithm is computationally costly because a proxy tree has to be constructed after any membership change, where a sink that joins or leaves a multicasting group causes a proxy to be added or removed. In other words, the proxy set has to be recomputed for any sink addition or deletion. The two on-line algorithms, namely, the *approximated on-line minimum Steiner tree* (ONMST) and the *enhanced ONMST* (E-ONMST), on the other hand, are not convenient for WSNs due to the large amount of overhead they both introduce in order for a new proxy or its neighbor to gather necessary information to construct a *Voronoi* diagram and reconfigure the subgraph surrounding it whenever a proxy changes due to target and sink mobility. Recall that each sensor has only partial knowledge about its multicasting group, that is, its neighbors in the tree. To alleviate this problem, two distributed heuristic-based algorithms, namely, the *shortest-path* (SP) *based* algorithm and the *spanning-range* (SR) *based* algorithm, were suggested.

The shortest-path based algorithm allows a proxy to join or leave a proxy tree and has three phases: *presearching, finding the closest node*, and *node join*. Let P_n be the proxy of a sink that wants to join the proxy tree. In the *presearching* phase, the proxy identifies the location of the current source proxy using an index-based technique, in which some sensors, called *index nodes*, maintain the locations of the sources [60]. The proxy P_n queries the appropriate index node to learn about the location of the source proxy and sends a *join_req* message to the root (or source proxy) of the proxy tree. When the root receives the *join_req* message, it uses geographical routing to send back a *join_req* message to P_n. This message will be forwarded until it reaches the closest sensor, denoted by P_j, to P_n. Then, P_n determines the closest sensor P_i by flooding a discovery message within a circle of radius equal to the distance $d(P_n, P_j)$ between itself and P_j. In the *finding the closest node* phase, upon receiving the replies from the sensors within the circle, P_n identifies P_i and sends a confirm message to P_i. In the *node*

join phase, upon receiving the confirm message, P_i adds P_n to the proxy tree and reconfigures the resulting subtree including itself and its neighbors into a *full Steiner tree* (FST). As can be observed, the new sink proxy P_n joins the proxy tree by attaching itself to the closest sensor in the tree. Leaving the tree depends on whether P_n is a leaf in the tree or not. If it is a leaf, it sends a *leave_req* message to its parent. If its parent is a Steiner node and has only two neighbors, these two neighbors directly connect to each other and the Steiner node is removed. However, if P_n is not a leaf, it marks itself as a Steiner node and stays in the tree. When a sink/source becomes far away from its proxy, the tree should be reconfigured. Specifically, assume that P_n' is the new proxy that is closer to the sink/source. First, P_n' sends a *migrate_req* message to the old proxy P_n, which in turn adds a temporary edge between P_n' and its parent, denoted by X, and leaves the tree. Similar to a new proxy joining the proxy tree, P_n' identifies its closest sensor P_i. If P_i is not X, P_n' connects to P_i and disconnects from X.

The spanning-range based algorithm improves the shortest-path based algorithm by using flooding to locate itself in the proxy tree. Flooding can degrade the performance of the data dissemination protocol. Specifically, the spanning-range based algorithm assigns a certain *spanning range* to each subtree whose nodes are within the range. Moreover, each node in the tree can decide the spanning range of each of its children using a few simple rules for space decomposition based on the locations of a child and its parent. When a mobile sink decides to join the multicasting tree, its proxy P_n sends a *join_req* message to the source proxy P. The source decides whether to add P_n as its immediate child or forwards its *join_req* message to one of its children whose spanning range covers P_n. This decision is based on the location of P_n with respect to the spanning ranges of the children of the source P computed by P. If P_n cannot be added as a direct child of P, the child receiving the *join_req* message will act exactly as its parent P to decide whether to add P_n as its direct child or forward the *join_req* message to the appropriate child. This process repeats until the sink proxy P_n gets attached as a child to one of the subtrees of the proxy tree.

As a result of sink mobility, the procedure of adding a new sink proxy and deleting the old one for the mobile sink requires migration of the role of a sink proxy from one sensor to another and attaching the new proxy to the proxy tree. The new proxy, say P_n', sends a *migrate_req* to the old proxy P_n, which in turn removes itself from the tree if it is a leaf and sends an *add_req* message to its parent in order to add P_n' to the proxy tree. Similarly, when a source is far away from its source proxy, the proxy tree needs to be reconfigured as the root has changed. Thus, the new spanning ranges are computed by the new root (or source proxy) and forwarded to all of its children. Similarly, each of these children will check whether the spanning ranges of their own children need any maintenance.

4.4.6 QoS Based Protocols

In addition to minimizing energy consumption, it is also important to consider QoS requirements in terms of delay, reliability, and fault tolerance in routing and

data dissemination in WSNs. This section discusses several QoS based routing and data dissemination protocols that help find a balance between energy consumption and QoS requirements.

4.4.6.1 Trade-Off between Energy Savings and Delay. In WSNs, the delay in the transmission of sensed data depends on the transmission time because there is no queuing delay and the propagation and processing delays are negligible compared to the transmission time. On one hand, given N deployed sensors, transmitting sensed data directly to the sink requires a total delay of N units when the sensors transmit one at a time to the sink. This delay can be lowered to $\log N$ units if the sensors are allowed to transmit their sensed data simultaneously to the sink using a binary scheme. However, the energy consumed in data transmission is proportional to the square of the transmission distance between the sending and receiving sensors. For this reason, direct transmission will incur significant energy consumption. On the other hand, minimizing energy introduces longer delay if sensed data have to be sent over short distances. Hence, there is a trade-off between energy consumption and transmission delay.

To account for the delay cost per round of data gathering, Lindsey et al. [40,41] proposed an *energy* × *delay* metric and two data gathering schemes to trade-off between energy and delay. By minimizing *energy* × *delay*, it is possible to achieve acceptable delay for those time-critical applications while reducing energy consumption in sensors, thus extending the network lifetime. Note that simultaneous wireless communications among pairs of sensors is possible if the sensors are CDMA capable, but low interference between those transmissions will occur. In this case, a *binary chain-based scheme*, which is an updated version of the PEGASIS protocol, can be used. However, the *energy* × *delay* cost depends on the sensor distribution in the sensor field. Recall that PEGASIS constructs a chain of sensors in which each sensor receives sensed data from its neighbor in the chain, fuses them with its own data, and forwards them to its neighbor. Assume that the neighboring sensors are equidistant from each other and this distance is equal to d. Then, there are $N/2$ sensors transmitting their sensed data at distance d. According to PEGASIS, the receiving sensors will fuse their own sensed data with the data they have received and become active in the next level of the tree as shown in Fig. 4.15. Therefore, only $N/4$ sensors will be transmitting their sensed data, but at distance $2d$. The same process repeats until the fused data is received by the last sensor, which performs data fusion with its own data and transmits the fused data to the sink. Therefore, the total energy cost for this binary chain-based scheme is proportional to

$$N/2 \times d^2 + N/4 \times (2d)^2 + N/8 \times (4d)^2 + \ldots + 1 \times (N/2 \times d)^2,$$

which is approximated by $N^2/2 \times d^2$ provided that we consider energy consumption in data transmission to the sink.

If the sensors are not CDMA capable, the use of the binary chain-based scheme would introduce a considerable amount of interference. In order to solve

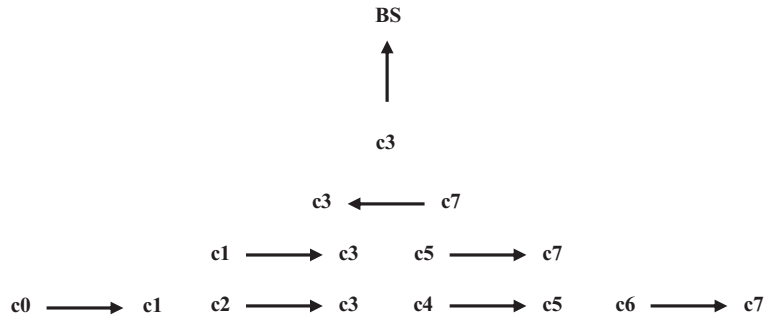

Fig. 4.15 Data gathering in a chain-based binary scheme.

this problem, Lindsey et al. [40,41] proposed a *three-level chain-based scheme*, in which simultaneous transmissions among pairs of spatially separated sensors are possible. As its name indicates, this scheme constructs a three-level hierarchy where each level contains a few groups and each group promotes one sensor to the next level. Figure 4.16 illustrates an example of this scheme. In this example, a set of N sensors is split into G groups, each of which has N/G successive sensors. The value of G is computed based on the number of sensors in the network and the size of the sensor field. Each group will promote one sensor to be active in the next level. Thus, the selected G sensors from the first level will be split into two groups in the second level and a sensor is promoted from each group to have two sensors in the third level. One of these two sensors will be promoted to the last level from which it will transmit the fused data to the sink. Figure 4.16 shows a chain of 100 non-CDMA sensors in the first level of the three-level hierarchy. It was found that for a $100\,\text{m} \times 100\,\text{m}$ sensor field and a network with 100 sensors the best balance between energy and delay is obtained for a value of $G = 10$. This means that only 10 simultaneous transmissions can take place and data fusion occurs at each sensor except the end ones in each level. Note that the leader that will transmit to the sink changes from one round to another in order to balance the load among the sensors.

4.4.6.2 Trade-Off between Energy Savings and Robustness. One of the main requirements of a network for some applications is *functionality*. In other words, the network should remain functional in spite of the occurrence of sensor failures. For this purpose, it is necessary for routing protocols to be fault tolerant (or robust) and at the same time guarantee energy efficiency. To provide robustness, conventional approaches attempt to control the transmission power while maintaining connectivity between sensors and use multipath routing, in which multiple disjoint or partially disjoint communication paths between source sensors and the sink are used. Different from these approaches, Krishnamachari et al. [42] proposed an approach that uses a single-path routing scheme with higher transmission power, but can achieve robustness against sensor failures.

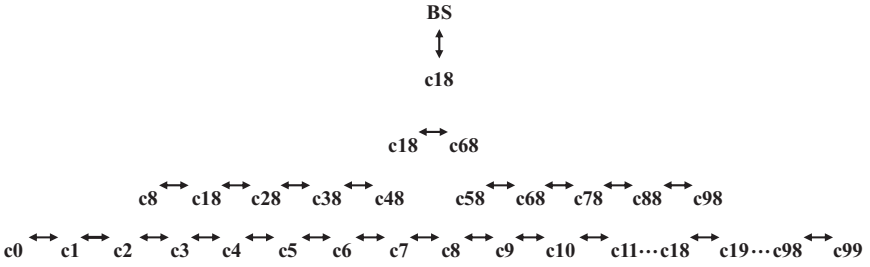

Fig. 4.16 Chain-based three-level scheme for non-CDMA sensors.

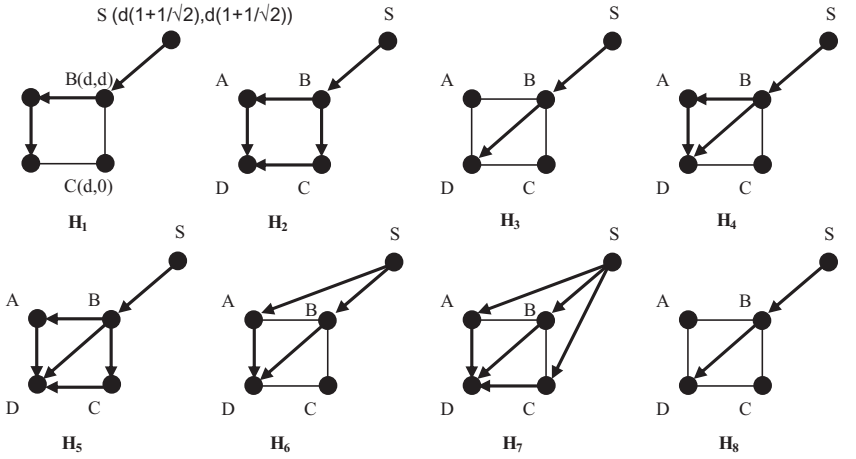

Fig. 4.17 Alternate routing configurations.

Using this approach, for each routing scheme H that routes information from a source sensor to a receiver, an energy metric equal to $m_H R_H^{\alpha}$ is assigned, where R_H is the minimum common transmission radius required for this routing scheme, m_H is the number of transmissions required for the information to reach the receiver, and α stands for the path-loss exponent. The approach assumes that any intermediate sensor can fail independently of other sensors with probability p while the source and the destination are perfectly reliable. The robustness metric Π_H associated with the routing scheme H is equal to the probability that a message initiated by a source sensor reaches the sink. Figure 4.17 shows different routing schemes for routing information from the source sensor S to the destination sensor D while Table 4.3 gives the energy and robustness measures for each of these routing schemes with $\alpha \in \{2, 4\}$.

Given the above assumption that the source and destination sensors are perfectly reliable, the routing scheme H_8 is the most robust one. However, direct

TABLE 4.3 Energy and Robustness Measures for Alternate Routing Configurations

Routing scheme H	E_H ($\alpha = 2$)	E_H ($\alpha = 4$)	Robustness Π_H
H_1	$3d^2$	$3d^4$	$(1\text{-}p)^2$
H_2	$4d^2$	$4d^4$	$(1\text{-}p)(1\text{-}p^2)$
H_3	$4d^2$	$8d^4$	1-p
H_4	$6d^2$	$12d^4$	1-p
H_5	$8d^2$	$16d^4$	1-p
H_6	$10.2d^2$	$35d^4$	$1\text{-}p^2$
H_7	$13.7d^2$	$46.6d^4$	$1\text{-}p^3$
H_8	$5.2d^2$	$34d^4$	1

transmission between the source and destination is highly costly in terms of energy consumption. Thus, there must be a trade-off between robustness, which has to be maximized, and energy consumption, which has to be minimized. Let H_i and H_j be two routing schemes. H_i *dominates* H_j if ($\prod_{H_i} \geq \prod_{H_j}$ and $E_{H_i} < E_{H_j}$) or ($E_{H_i} \leq E_{H_j}$ and $\prod_{H_i} > \prod_{H_j}$). The routing schemes that are not dominated by any other routing scheme form the *Pareto set* and are called *Pareto optimal*. According to Table 4.3, it is clear that $\{H_1, H_3, H_8\}$ is the Pareto set. Note that those Pareto optimal routing schemes provide single-path routes. Moreover, although the routing scheme H_8 uses direct transmission, it consumes less energy than multipath routing schemes H_6 and H_7. This result means that when robustness and energy efficiency are the main concerns, single-path routing outperforms multipath routing under the assumption of perfectly reliable source and destination sensors.

4.4.6.3 Trade-Off between Traffic Overhead and Reliability. While single-path routing routes a sensed data packet from a source sensor to the sink through a sequence of intermediate sensors acting as forwarders, multipath routing routes the same data packet via multiple paths between source and destination sensors. The former scheme is sensitive to the failure of intermediate sensors whereas the latter increases the reliability of data transmission, but yields much overhead. Dulman et al. [43] suggested that a variant of multipath routing be used, where a data packet is split into k subpackets of equal size with added redundancy and sent over k available disjoint paths. In order to construct the original data packet at the destination sensor, only a smaller number of subpackets are needed. The amount of redundancy that should be added for the split message transmission is determined based on the number of successful paths. Let S_k be a random variable representing the number of successfully delivering paths. It is clear that S_k is upper bounded by k, that is, $S_k \leq k$. In fact, all the generated paths are disjoint and the experiment corresponding to transmitting a data packet from a source sensor to a destination sensor can be viewed as a repeated Bernoulli experiment. For the ith path, if the transmission succeeds, this subrun

is assigned 1; otherwise, it is assigned 0. The value of S_k is the sum of the values assigned to the k subruns as there are k disjoint paths. Thus, the expected number of successful delivering paths can be calculated as

$$E(S_k) = \sum_{i=1}^{k} p_i,$$

where p_i is the probability of successfully delivering a message to a destination using the ith path. In order to compute a good estimation for the value of E_k for a given bound α representing the overall probability of successfully reconstructing the transmitted message at the destination such that $P(S_k \geq E_k) \geq \alpha$, Dulman et al. [43] approximated the repeated Bernoulli experiment by a standard distribution $N(\mu, \sigma)$, where the mean is given by

$$\mu = E(S_k) = \sum_{i=1}^{k} P_i$$

and the standard deviation is calculated as

$$\sigma^2 = \sum_{i=1}^{k} p_i (1 - p_i).$$

Given that the total number of subpackets depends on the degree k of multipath routing, a given pair $(k, \{p_1, \ldots, p_k\})$ produces a different normal distribution, $N(\mu(k), \sigma(k))$. To solve this problem, the random variable S_k is transformed into $S_k^* = (S_k - \mu)/\sigma$, which is $N(0, 1)$ distributed. However, the values of the bound x_α are known (see Table 4.1 [43]) for any given α such that $P(S_k^* \geq x_\alpha) \geq \alpha$ is satisfied. As a result, $S_k^* = \dfrac{S_k - \mu}{\sigma} \geq x_\varepsilon$ implies that $S_k \geq x_\alpha \times \sigma + \mu$, and hence we have the following probability

$$P(S_K \geq x_\alpha \times \sigma + \mu) \geq \alpha.$$

By equating this probability with $P(S_k \geq E_k) \geq \alpha$, we obtain an approximation of E_k for a given bound α; that is,

$$E_k = \max(\lfloor x_\alpha \times \sigma + \mu \rfloor, 1).$$

By using the previously computed values of μ and σ, the value of E_k is given by

$$E_k = \max\left(\left\lfloor x_\alpha \times \sqrt{\sum_{i=1}^{k} p_i (1 - p_i)} + \sum_{i=1}^{k} p_i \right\rfloor, 1\right),$$

which gives a good estimation of the number of successfully delivering paths for a given bound α.

4.4.7 Heterogeneity-Based Protocols

All the routing and data dissemination protocols discussed so far assume a homogeneous network architecture, in which all sensors have the same capabilities in terms of battery power, communication, sensing, storage, and processing. Recently, there has been an interest in heterogeneous sensor networks, especially for real deployment. For example, Intel has deployed a pilot application of heterogeneous sensor networks. The proposed architecture uses two types of sensors: the sensors attached to pumps and motors in a fabrication plant have no energy constraint (i.e., line-powered sensors), whereas the others are battery-powered sensors in order to reduce the installation cost and complexity. Those battery-powered sensors have limited lifetime, and hence should use their available energy efficiently by minimizing their potential of data communication and computation.

Another real deployment of heterogeneous sensor networks can be found in [61]. This study demonstrated that CrossBow Mica and iPAQ motes can be integrated together in the same architecture. Since Mica motes use very little power and perform complex computation, it is more efficient to deploy them for sensing only. The iPAQ motes are suitable for data fusion because they consume more power and perform computation quickly. It has been shown that network lifetime can be extended provided that an intelligent assignment of tasks on the heterogeneous sensors is guaranteed.

Heterogeneous networks are attractive as they can potentially extend network lifetime, which is defined as the *time to the first sensor death*, and improve reliability. For both energy efficiency and cost effectiveness, this type of network requires that a large number of inexpensive sensors perform sensing while a few expensive sensors provide in-network processing and data forwarding to the sink. This section discusses how heterogeneity can be used to extend network lifetime and present a few routing and data dissemination protocols for heterogeneous WSNs.

4.4.7.1 Benefits of Heterogeneity in Wireless Sensor Networks. A network consists of two main components: sensors and communication links between them. Hence, heterogeneity can be introduced at these two levels, thus leading to network deployment with *energy heterogeneity* and/or *link heterogeneity*. Yarvis et al. [50] proposed a three-layer architecture for heterogeneous WSNs. In this architecture, the top layer contains only one sink that receives sensed data and analyze them. The second layer includes sensors with no energy constraint. These sensors, called *line-powered* sensors, have unlimited energy resources by connecting them to a wall outlet. The third layer contains battery-powered sensors that are 1-hop away from line-powered sensors. The rationale behind this architecture is that the sensors closer to the sink in a multihop sensor network with many-to-one delivery consume more energy than all other sensors in the network, and thus should be line powered. As observed, this three-layer architecture forms a dominating tree, where each battery-powered sensor communicates with the sink via only line-powered sensors to transmit its sensed data.

There is no communication among battery-powered sensors in order to save their energy, and hence no battery-powered sensor can play the role of a data forwarder on behalf of other sensors. Obviously, there should be a sufficient number of line-powered sensors. If we assume that most of the energy consumption is due to data communication, it can be easily proved that this dominating tree of line-powered sensors rooted at the sink can extend network lifetime by at least $nS_{e2e}/mS_{\text{link}}$ compared to a network architecture without energy heterogeneity, where n is the network size, m is the number of sensors within the radio range of the sink, S_{e2e} is the average end-to-end delivery rate, and S_{link} is the link success rate in the vicinity of the sink [50].

In addition to heterogeneous sensors with regard to their energy, the communication links between the sensors can be heterogeneous as well. To realize link heterogeneity, some sensors need to have high-quality links to the sink; that is, long-distance highly reliable communication links, for example, 802.11 type connectivity. This link characteristic will reduce the average number of hops required for a packet transmitted by a battery-powered sensor to reach the sink. Thus, these high-quality links (or backhaul links) help decrease the end-to-end delay and energy consumption while increasing the end-to-end delivery rate. In other words, the objective of adding heterogeneous links to the network is to increase the rate of successful packet delivery to the sink. The sensors followed by a highly reliable hop to the sink are called *backhaul sensors*. For example, assume that the heterogeneous sensors are deployed in an $m \times n$ Manhattan grid. It is easy to check that the length of the shortest path from a sensor located at location (i, j) to a sink that is adjacent to the midpoint of one edge can be calculated as

$$d\{(i, j), s\} = \left| \frac{m-1}{2} - i \right| + (j+1).$$

If we consider backhaul sensors, each of which is one hop away from the sink, the length of the shortest path from a sensor located at location (i, j) to a sink via a backhaul sensor located at location (k, l), which is denoted by $b(k, l)$, is calculated as

$$d^{b(k,l)}_{((i,j),s)} = |k-i| + |l-j| + 1.$$

Thus, the length of the delivery path for a sensor located at location (i, j) is given by

$$d^{*}_{((i,j),s)} = \min\left\{ d_{((i,j),s)}, d^{b(k,l)}_{((i,j),s)} \right\}$$

over all possible backhaul sensors. Therefore, the sum of all shortest paths from each sensor to the sink is given by

$$S_{m,n} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} d^{*}_{((i,j),s)}.$$

Hence, there is an optimal deployment of backhaul links for a given $m \times n$ Manhattan grid that minimizes $S_{m,n}$. The benefits of heterogeneity depend on the number of backhaul sensors and their locations. The maximum benefit is obtained when each sensor is one hop away from either a backhaul sensor or the sink. In this case, the end-to-end success rate gets closer to the link success rate. Furthermore, the benefit increases with the number of backhaul sensors. For an arbitrary topology, the optimal deployment of heterogeneous resources (energy and links) corresponds to a tree rooted at the sink. More specifically, the maximum benefit of heterogeneous resources, such as energy heterogeneity and link heterogeneity, depends on the shape, size, and density of the network. For more information about optimal resource deployment, the interested reader is referred to Ref. [50].

4.4.7.2 Information-Driven Sensor Query. An interesting problem in heterogeneous WSNs is how to maximize information gain and minimize detection latency and energy consumption for target localization and tracking through dynamic sensor querying and data routing, which is addressed in Ref. [41]. To improve tracking accuracy and reduce detection latency, communication between sensors is necessary and consumes significant energy. In order to conserve power, only a subset of sensors need to be active when there are interesting events to report in some parts of the network. The choice of a subset of active sensors that have the most useful information is balanced by the communication cost needed between those sensors. Useful information can be sought based on predicting the space and time interesting events would take place.

If x is the parameter representing the target position that we want to estimate, the *belief* that is defined as a representation of the current *a posteriori* distribution of x given a set of measurements $z_1, \ldots, z_N$ is calculated as

$$p(x|z_1, \ldots, z_N)$$

and the expectation value of this distribution is considered as the *estimate* and is given by

$$\bar{x} = \int xp(x|z_1, \ldots, z_N)\,dx,$$

where

$$z_i = \frac{a}{\|x_i - x\|^{\alpha/2}} + w_i$$

with a being a random variable representing the amplitude of the target, α a known attenuation coefficient, w_i a zero mean Gaussian random variable, and $\|\cdot\|$ the Euclidean norm. It is assumed that each sensor, s_i, is aware of its own location, $x_i \in \mathbf{IR}^2$. Since the belief is calculated based on measurements from several sensors, there will be a cost to collect the information. Therefore, it is

necessary to select a subset of sensor measurements providing good information to build a belief state while minimizing the cost of communicating those measurements to a single sensor. To assess the information provided by a sensor measurement to a belief state, Chu et al. [44] introduced a measure called *information content*.

Chu et al. [44] proposed an *information-driven sensor querying* (IDSQ) protocol to optimize sensor selection. In the IDSQ protocol, the first step is to select a sensor l as a leader from a cluster of N sensors. This leader will be responsible for selecting optimal sensors based on some information utility measure, for example, the *geometric measure*, that is,

$$\psi(p_X) = (x_j - \hat{x})^T \sum{}^{-1} (x_j - \hat{x})$$

and the Mahalanobis distance of the sensor under consideration to the current position estimate of the target, and requesting data from them. The Mahalanobis distance measures the distance to the center of the error ellipsoid, normalized by the covariance Σ of the distribution $p_X(x)$. The leader l is supposed to have knowledge of certain characteristics $\{\lambda_i\}_{i=1}^N$ of the sensors, for example, their locations, $\lambda_i = x_i$. Each sensor that is not a leader will wait for a query from the leader sensor. When it is queried, a sensor processes its measures and sends the queried information back to the leader. When a target is within the range of the cluster of sensors, the sensor leader l becomes activated. This activation can be done when, for example, the amplitude reading at the leader is higher than a certain threshold. Then, the leader l computes a representation of the belief state using its own measurement, $p(x|z_l)$, and keeps track of the sensors' measurements that have been incorporated into the belief state, $U = \{l\} \subset \{1, \ldots, N\}$. Based on the quality of the belief, which can be assessed using some goodness measure, the leader may finish processing or continue with sensor selection. In case the belief is not good enough, the leader runs its sensor selection algorithm based on the belief state $p(x|\{Z_i\}_{i\in U})$ and the sensor characteristics $\{\lambda_i\}_{i=1}^N$. When the leader selects a sensor j from the set $\{1, \ldots, N\}\backslash U$ to request data from j, it sends a query to j and waits for the queried data. Upon receiving the information z_j from j, the leader updates its current belief state $p(x|\{z_i\}_{i\in U})$ with z_j, thus leading to a new belief state $p(x|\{z_i\}_{i\in U} \cup z_j)$. Then, it updates the set of sensors that have been incorporated so far; that is, $U = U \cap \{z_j\}$, and check again its belief state as to whether it is good enough. Note that the leader queries only a subset of sensors to obtain the most useful information for it to build its belief state. This intelligent selection helps the sensors save their energy if they do not have pertinent information about a target to communicate to the leader. In Section 4.4.7.3, we describe how the query and the information are routed between a querying sensor (or leader) and the queried sensor using an algorithm, called *constrained anisotropic diffusion routing* (CADR) [60].

4.4.7.3 Constrained Anisotropic Diffusion Routing. By using CADR [60], the selection of the optimal routing path is dynamic and is guided by the

composite objective function that considers the information utility and the actual cost of bandwidth and latency. This composite objective function, M_c, is defined as

$$M_c(\lambda_l, \lambda_j, p(x|\{z_i\}_{i\in U})) = \gamma M_u(p(x|\{z_i\}_{i\in U}), \lambda_j) - (1-\gamma) M_a(\lambda_l, \lambda_j),$$

where $M_u(p(x|\{z_i\}_{i\in U}), \lambda_j)$ represents the information utility function. The parameter $M_a(\lambda_l, \lambda_j)$ stands for the cost of the bandwidth and latency of information communication between sensor j and sensor l, and $0 \le \gamma \le 1$ is a trade-off parameter that balances the contribution from the two terms. Now, let us discuss different cases for query and information routing using CADR.

Case 1: Routing with Global Knowledge of Sensor Positions. In this case, a querying sensor is aware of the locations of all sensors in the network. The best next sensor to select is the one closest to the optimal position x_0; that is,

$$x_0 = \arg_x [\nabla M_c = 0],$$

where ∇M_c stands for the gradient of the composite objective function M_c. It is possible to establish a routing path toward the potentially best sensor along which the measurement from the sensor closest to the optimal position is sent back to the querying sensor. Given this global knowledge of sensor positions, the routing path is optimal.

Case 2: Routing without Global Knowledge of Sensor Positions. In this case, the information query routing is based on localized decisions by individual sensors that consider the regions in the sensor field, where the constraints imposed by the composite objective function M_c are met. Furthermore, since the belief state undergoes updates along the routing path, the function M_c is also updated. The current routing sensor k that holds the information query and is located at x_k selects one of its neighbors $\hat{j}$ as the best next sensor that maximizes the objective function M_c. Formally, this local selection can be expressed as

$$\hat{j} = \arg_j \max[M_c(x_j)], \forall j \ne k.$$

Also, the current routing sensor k can choose the next best one $\hat{j}$ among its neighbors that is located in the direction of ∇M_c, thus satisfying

$$\hat{j} = \arg_j \max\left(\frac{(\nabla M_c)^T (x_k - x_j)}{|\nabla M_c||x_k - x_j|} \right).$$

Another alternative to select the next best sensor is to first determine the direction toward the minimum objective function at any routing step by solving

$$x_0 = \arg_x [\nabla M_c = 0],$$

which allows computing $x_0 - x_k$ that corresponds to the direction toward the optimal position x_0. Then, the next routing sensor can be selected based on the distance

$$d = \beta \nabla M_c + (1-\beta)(x_0 - x_k),$$

where β is a parameter that is defined as a function of the distance between the current and optimal sensor positions, that is, $\beta = \beta(|x_0 - x_k|)$. Hence, for a large distance d, it would be better to follow the gradient of the objective function. Otherwise, it is more efficient to go toward the minimum rather than following the gradient ascent. As can be seen, this routing alternative chooses the routing direction based on the distance from the optimal position.

The evaluation of the composite objective function and its derivatives requires that a query be sent together with the information on the current belief state. This information should be enough to update the belief state incrementally based on local sensor measurement. In case the Mahalanobis distance is used to quantify the information utility, the triplet $\left\{\hat{x}, x_q, \hat{\Sigma}\right\}$ has to be sent together with the query, where $\hat{x}$ is the current state of the estimated location of the target, x_q is the location of the querying sensor, and $\hat{\Sigma}$ is the current estimate of the uncertainty covariance of the target position. Similarly, a routing path toward the potentially best sensor can be established, along which the measurement from the sensor closest to the optimal position is sent back to the querying sensor. However, this routing path is locally optimal given the greedy nature of the routing algorithm. Moreover, the information provided by the sensors along the path improves incrementally toward the global optimum given that the information utility objective function is monotonically increasing.

4.4.7.4 Cluster-Head Relay Routing. The *cluster head relay* (CHR) protocol [45] uses two types of sensors to form a *heterogeneous* network with a single sink: a large number of low-end sensors, denoted by *L-sensors*, and a small number of powerful high-end sensors, denoted by *H-sensors*. Both types of sensors are static and aware of their locations using some location service. Moreover, those L- and H-sensors are uniformly and randomly distributed in the sensor field. The CHR protocol partitions the heterogeneous network into groups of sensors (or *clusters*), each being composed of L-sensors and led by an H-sensor.

Within a cluster, the L-sensors are in charge of sensing the underlying environment and forwarding data packets originated by other L-sensors toward their cluster head in a multihop fashion. The H-sensors, on the other hand, are responsible for data fusion within their own clusters and forwarding aggregated data packets originated from other cluster heads toward the sink in a multihop fashion using only cluster heads. While L-sensors use short-range data transmission to their neighboring H-sensors within the same cluster, H-sensors perform long-range data communication to other neighboring H-sensors and the sink. Because of their different functioning modes, H-sensors have more powerful resources than L-sensors. As any cluster-based routing protocol, CHR has three phases: cluster formation, intracluster routing, and intercluster routing. At the beginning, the sink broadcasts its location to all H-sensors in the network. These H-sensors advertise their IDs and locations through Hello messages to the L-sensors with a certain random delay in order to avoid collisions between those messages. Upon receiving those Hello messages, each L-sensor selects an H-sensor as its *primary cluster head* based on the strength of the received signal. More specifically, each L-sensor chooses the closest H-sensor as its cluster head. However, in the presence of obstacles, the network is modeled by a *Voronoi* diagram where the nuclei of *Voronoi* cells are the cluster heads. In addition, each L-sensor stores the IDs and locations of the other H-sensors that will serve as backup cluster heads in case of a primary cluster head failure. Up to now, each sensor belongs to only one cluster.

Once an L-sensor selects an H-sensor, it starts sending its sensed data to the H-sensor. If an L-sensor does not have any sensed data to send after T seconds of deployment, it sends a specific location packet to its H-sensor, including its physical location. Therefore, after T seconds, each H-sensor has learned the locations of all L-sensors belonging to its cluster. For each L-sensor, the corresponding H-sensor computes two routes based on the locations of its L-sensors: one is called an *optimal* route, which can be formed based on energy consumption, hop count, or any other metric, whereas the second is called a *suboptimal* route. Then, each H-sensor informs its members of those optimal and suboptimal routes. For this purpose, each H-sensor first divides its cluster into sectors. The number of sectors depends on the number of sensors in the cluster and should be a trade-off between the number of broadcast messages and the message length. Before sending those two routes to each of the L-sensors in its cluster, the H-sensor sends a short message specifying the ID of the sector whose L-sensors should consider the routes being disseminated. Then, the H-sensor broadcasts a long message including the two routes for each L-sensor in its cluster. Note that only the L-sensors in the sector whose ID is recently advertised will be able to receive this long message. If both routes for a given sensor s are not available (e.g., the next hops failed), this sensor, s, broadcasts its packet to its neighbors. If one of the neighbors, say s', replies with an acknowledgment that it knows a route to the sink, s forwards its data to s', which in turn forwards the data to the sink. Otherwise, s becomes disconnected from the sink.

Each cluster head advertises its IDs and locations to its neighbor cluster heads. To send its data to the sink, a cluster head forwards its packet to the cluster head whose *Voronoi* cell is crossed by a straight line connecting cluster head and the sink. Such a *Voronoi* cell is called a *relay cell*. Specifically, the packet will be forwarded from the source cluster head (or simply *source*) to the sink along the cluster heads whose *Voronoi* cells are relay cells. The intercluster routing is similar to a source-initiated routing, in which a route is specified by the source of the message (a cluster head in this case). To enforce this route, the source specifies in the header of the packet the relay cell list as well as the source ID, sink ID, and session ID. The pair (source ID, session ID) uniquely identifies a data dissemination session. To guarantee data delivery, the current cluster head waits for an acknowledgment within a timeout. If it does not hear this acknowledgment, it resends the packet to the same next cluster head. In case the transmission fails again, the current cluster head uses a backup path, which is constructed using the same approach. In other words, this cluster head identifies the relay cells with respect to the sink. If the next relay cell is the one that has failed, the cluster head uses a detoured path to avoid the cell. Otherwise, the new set of relay cells will be used as the actual forwarding path to the sink.

4.4.8 Comparisons

Although we have classified a sample of routing and data dissemination protocols for WSNs according to our taxonomy, it should be mentioned that some of those protocols fit into more than one class. For example, PEGASIS [18] uses data aggregation and helps find a balance between energy and delay [40,41]. Table 4.4 shows the similarities between the protocols surveyed in this chapter with respect to the classification criteria used in the taxonomy.

It is worth mentioning that the transmission of sensed data to the sink takes place in one of the following forms: on-demand, continuous, triggered, and hybrid. In other words, there are four potential data delivery models. For some sensing applications, the source sensors send their sensed data continuously to the sink. For example, in a temperature-monitoring application, the sensors send their data to the sink in a continuous manner without looking at the values obtained. To make their task more energy efficient, the sensors can send their data only when the value of the temperature is above or below a certain threshold. In other words, data transmission is triggered by an event that is based on the threshold. Also, data transmission can be initiated in an on-demand fashion in which the sink requests data from the source sensors by sending them specific queries. These three forms of data transmission can also take place within the same sensing application, that is, in the hybrid form. Moreover, data transmission can be either *broadcast* throughout the network or *unicast* to specific sensors based on some criteria. Although *data delivery models* deal with data delivery from an application traffic perspective [2], routing protocols can also be classified based on the type of data delivery models being used by sensing applications.

TABLE 4.4 Comparison of Routing and Data Dissemination Protocols for WSNs

Classification Criteria	Protocols
Location awareness	GAF, GEAR, Span, TBF, BVGF, GeRaF, MECN, SMECN, PEGASIS, Quorum and home agent-based information dissemination, Joint mobility and routing, TTDD, SEAD, Dynamic proxy tree based data dissemination, Energy-robustness trade-off, IDSQ, CADR, CHR
Network layering	GAF, LEACH, PEGASIS, TEEN, APTEEN, Cougar, EAD, Data MULEs, TTDD, SEAD, Dynamic proxy tree based data dissemination, Energy-delay trade-off, IDSQ, CADR, CHR
In-network processing	LEACH, PEGASIS, TEEN, APTEEN, SPIN, Directed diffusion, Rumor routing, Cougar, ACQUIRE, EAD, Information directed routing, TTDD, Energy-delay trade-off, CHR
Data centricity	GEAR, TEEN, APTEEN, SPIN, Directed diffusion, Rumor routing, Cougar, ACQUIRE, EAD, Information directed routing, Quorum and home agent-based information dissemination
Multipath	TBF, SPIN, Directed diffusion, Sensor-disjoint multipath, Braided multipath, N-to-1 multipath discovery, Energy-robustness trade-off, Overhead-reliability trade-off
Mobility	GAF, TBF, MECN, SMECN, Joint mobility and routing, Data MULEs, TTDD, SEAD, Dynamic proxy tree-based data dissemination.
Quality-of-Service	PEGASIS, TEEN, APTEEN, SPIN, Directed diffusion, Information directed routing, Sensor-disjoint multipath, Braided multipath, N-to-1 multipath discovery, Data MULEs, TTDD, Energy-delay trade off, Energy-robustness trade-off, Overhead-reliability trade off, IDSQ, CADR
Heterogeneity	Data MULEs, IDSQ, CADR, CHR
Energy awareness	GAF, GEAR, Span, MECN, SMECN, LEACH, PEGASIS, TEEN, APTEEN, SPIN, Directed diffusion, Cougar, EAD, Sensor-disjoint multipath, Braided multipath, Joint mobility and routing, Data MULEs, TTDD, SEAD, Energy-delay trade off, Energy-robustness trade off, IDSQ, CADR, CHR

4.5 SUMMARY AND FUTURE DIRECTIONS

One of the main challenges in the design of protocols for WSNs is energy efficiency due to the scarce energy resources of sensors. The ultimate objective behind the protocol design is to keep the sensors operating for as long as possible, thus extending the network lifetime. In particular, routing and data dissemination

is a vital task in WSNs, and thus the design of routing and data dissemination protocols should be specially taken care of. To accomplish their monitoring operation appropriately, the sensors in a network need to collaborate with each other by acting as forwarders of data and control messages on behalf of others. Therefore, it is necessary for the sensors to get involved in the communication between themselves during their operation. However, the energy consumption of the sensors is dominated by data transmission and reception. Specifically, the energy consumed by the sensors in processing (or computation) and sensing is negligible compared to that in data communication. Therefore, routing and data dissemination protocols designed for WSNs should be as energy efficient as possible to prolong the lifetime of individual sensors, and hence the network lifetime. On the other hand, there are other QoS requirements, for example, delay and fault tolerance, to name a few, which should also be considered in the protocol design. To meet such requirements, for example, to minimize delay and increase fault tolerance, some conflicts with the goal of guaranteeing energy efficiency could be introduced. Therefore, a reasonable trade-off should be established between energy efficiency and those QoS requirements.

This chapter surveyed a sample of routing and data dissemination protocols for WSNs based on our proposed taxonomy. This taxonomy takes into account several classification criteria, including location information, network layering and in-network processing, data centricity, path redundancy, network dynamics, QoS requirements, and network heterogeneity. For each of these categories, we have discussed a few example protocols. Our objective is to help the reader get a better understanding of those protocols and gain an insight into how to design efficient protocols that best meet the requirements of a sensing application.

We believe that two important related research directions should receive much attention from the community. While the first concerns the design of routing and data dissemination protocols for duty-cycled WSNs, the second is to consider three-dimensional (3D) sensor fields when designing such protocols. Most of the existing geographic routing and data dissemination protocols assume that all sensors in a network are awake during the forwarding activity. In real-world scenarios, however, the sensors switch between on and off states in order to save their limited energy. It is not even practical to keep a sensor awake all the time while it is active for some short periods of time. Moreover, some sensor failures may have a severe impact on the performance of the network. In case of a sensor failure, the network could be disconnected and partitioned into at least two noncommunicating subnetworks, and hence the existence of the whole network may become meaningless. Therefore, it is important to duty cycle the sensors so that they deplete their energy resources uniformly and slowly. Unfortunately, duty cycling may create a problem for routing a current message to the next hop while it is asleep. To get around this problem, the message can be either buffered until the next hop becomes awake or forwarded over the currently awake sensors. In the former case, a certain delay would be introduced, whereas in the latter case the number of hops may increase significantly, thus leading to considerable energy overhead. Nath and Gibbons [62] addressed this problem by providing a

scheduling algorithm that can be tuned to achieve a certain routing performance. It is more useful that all other routing and data dissemination protocols are designed to handle highly dynamic networks that experience time-varying connectivity due to sensor duty cycling. The challenge is how to duty cycle the sensors while guaranteeing good routing performance. It is also important to extend those protocols to k-covered WSNs, where each location in a sensoring field is covered by at least k sensors.

Although most of research work on WSNs, in particular, on routing and data dissemination, considered two-dimensional (2D) settings, where sensors are deployed on a planar field, there are some situations where the 2D assumption is not reasonable and the use of a 3D design becomes a necessity. In fact, 3D settings reflect more accurate network design for real-world applications. For example, a network deployed on the trees of different heights in a forest, in a building with multiple floors, or underwater, requires design in 3D rather than 2D space. Oceanographic data collection, pollution monitoring, offshore exploration, disaster prevention, and assisted navigation are all typical applications of underwater sensor networks [63]. Pompili et al. [63] proposed different deployment strategies for 2D and 3D communication architectures for underwater acoustic sensor networks, where the sensors are anchored at the bottom of the ocean for the 2D design and float at different depths of the ocean to cover the entire 3D region. Although some efforts have been devoted to the design of routing and data dissemination protocols for 3D sensing applications, we believe that these first-step attempts are in their infancy, and more powerful and efficient protocols are required to satisfactorily address all problems that may occur, including the ones prior to routing. Perhaps the most nontrivial conceptual problem is sensor deployment. Routing and data dissemination are strictly dependent on the sensor placement in a sensing field. A first question that arises is *How should sensors be placed in a 3D space so that the required quality of monitoring is satisfied*? More importantly, *How should connectivity between the sensors in a 3D space be guaranteed in order to provide a high-quality service of routing and data dissemination*? It has been proved that connectivity depends on coverage. More specifically, a network is connected if the network is configured to provide coverage and the radius of the communication range of the sensors is at least double the radius of their sensing range. Some studies have already considered sensing coverage and network connectivity in an integrated manner. From at least the above questions, it is clear that these three components, namely, sensing coverage, network connectivity, and routing and data dissemination should be studied together. We hope that such studies will be given more attention by researchers in their future work.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey", *Computer Networks*, vol. 38, no. 4, Mar. 2002, pp. 393–422.

[2] W. Wang, V. Srinivasan, and K.-C. Chua, "Using mobile relays to prolong the lifetime of wireless sensor networks", *Proceedings ACM/IEEE MobiCom'05*, Cologne, Germany, Sept. 2005, pp. 270–283.

[3] K. Akkaya and M. Younes, "A survey on routing protocols for wireless sensor networks", *Ad Hoc Networks*, vol. 3, no. 3, May 2005, pp. 325–349.

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

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

[6] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks", *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, Oct. 2002, pp. 660–670.

[7] F. Aurenhammer, "Voronoi diagrams—A survey of a fundamental data structure", *ACM Computing Surveys*, vol. 23, no. 3, Sept. 1991, pp. 345–405.

[8] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices", *IEEE Personal Communication Magazine*, vol. 7, no. 5, Oct. 2000, pp. 28–34.

[9] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad-hoc routing", *Proceedings ACM/IEEE MobiCom'01*, Rome, Italy, July 2001, pp. 70–84.

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

[11] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks", *Proceedings ACM MobiCom'01*, Rome, Italy, July 2001, pp. 85–96.

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

[13] B. Nath and D. Niculescu, "Routing on a curve", *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, Jan. 2003, pp. 155–160.

[14] G. Xing, C. Lu, R. Pless, and Q. Huang, "On greedy geographic routing algorithms in sensing-covered networks", *Proceedings ACM MobiHoc'04*, Tokyo, Japan, May 2004, pp. 31–42.

[15] M. Zorzi and R. R. Rao, "Geographic random forwarding (GeRaF) for ad hoc and sensor networks: Mutlihop performance", *IEEE Transactions on mobile Computing*, vol. 2, no. 4, Oct.–Dec. 2003, pp. 337–348.

[16] V. Rodoplu and T. H. Meng, "Minimum energy mobile wireless networks", *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, Aug. 1999, pp. 1333–1344.

[17] L. Li and J. Y. Halpern, "Minimum-energy mobile wireless networks revisited", *Proceedings IEEE ICC'01*, Helsinki, Finland, June 2001, pp. 278–283.

[18] S. Lindsey and C. S. Raghavendra, "PEGASIS: Power-efficient gathering in sensor information systems", *Proceedings IEEE Aerospace Conference*, vol. 3, Big Sky, MT, Mar. 2002, pp. 1125–1130.

[19] A. Manjeshwar and D. P. Agrawal, "TEEN: A routing protocol for enhanced efficiency in wireless sensor networks", *Proceedings IPDPS'01*, San Francisco, CA, Apr. 2001, pp. 2009–2015.

[20] A. Manjeshwar and D. P. Agrawal, "APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks", *Proceedings IPDPS'01*, San Francisco, CA, Apr. 2001, pp. 2009–2015.

[21] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks", *Proceedings ACM MobiCom'99*, Seattle, WA, Aug. 1999, pp. 174–185.

[22] J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks", *Wireless Networks*, vol. 8, no. 2/3, Mar.–May 2002, pp. 169–185.

[23] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks", *Proceedings ACM MobiCom'00*, Boston, MA, Aug. 2000, pp. 56–67.

[24] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking", *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, Feb. 2003, pp. 2–16.

[25] D. Braginsky and D. Estrin, "Rumor routing algorithm in sensor networks", *Proceedings ACM WSNA, in conjunction with ACM MobiCom'02*, Atlanta, GA, Sept. 2002, pp. 22–31.

[26] Y. Yao and J. Gehrke, "The Cougar approach to in-network query processing in sensor networks", *SGIMOD Record*, vol. 31, no. 3, Sept. 2002, pp. 9–18.

[27] N. Sadagopan, B. Krishnamachari, and A. Helmy, "The ACQUIRE mechanism for efficient querying in sensor networks", *Proceedings SNPA'03*, Anchorage, AK, May 2003, pp. 149–155.

[28] A. Boukerche, X. Cheng, and J. Linus, "Energy-aware data-centric routing in microsensor networks", *Proceedings ACM MSWiM, in conjunction with ACM MobiCom*, San Diego, CA, Sept. 2003, pp. 42–49.

[29] J. Liu, F. Zhao, and D. Petrovic, "Information-directed routing in ad hoc sensor networks", *Proceedings ACM WSNA'03, in conjunction with ACM MobiCom*, San Diego, CA, Sept. 2003, pp. 88–97.

[30] D. Liu, X. Hu, and X. Jia, "Energy efficient information dissemination protocols by negotiation for wireless sensor networks", *Computer Communications*, vol. 29, no. 11, July 2006, pp. 2136–2149.

[31] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks", *Proceedings ACM MobiHoc'01*, Long Beach, CA, Oct. 2001, pp. 251–254.

[32] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks", Mobile Computing and *Communications Review*, vol. 5, no. 4, Oct. 2001, pp. 10–24.

[33] W. Lou, "An efficient N-to-1 multipath routing protocol in wireless sensor networks", *Proceedings IEEE MASS'05*, Washington, DC, Nov. 2005, pp. 1–8.

[34] J. Luo and J.-P. Hubaux, "Joint mobility and routing for lifetime elongation in wireless sensor networks", *Proceedings IEEE INFOCOM'05*, vol. 3, Miami, FL, Mar. 2005, pp. 1735–1746.

[35] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: Modeling a three-tier architecture for sparse sensor networks", *Proceedings SNPA'03*, Anchorage, AK, May 2003, pp. 30–41.

[36] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, "TTDD: Two-tier data dissemination in large-scale wireless sensor networks", *Wireless Networks*, vol. 11, no. 1–2, Jan. 2005, pp. 165–175.

[37] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks", *Proceedings ACM/IEEE MobiCom'02*, Atlanta, GA, Sept. 2002, pp. 148–159.

[38] H. S. Kim, T. Abdelzaher, and W. H. Kwon, "Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks", *Proceedings ACM SenSys'03*, Los Angeles, CA, Nov. 2003, pp. 193–204.

[39] W. Zhang, G. Cao, and T. La Porta, "Dynamic proxy tree-based data dissemination schemes for wireless sensor networks", *Proceedings IEEE MASS'04*, Fort Lauderdale, FL, Oct. 2004, pp. 21–30.

[40] S. Lindsey, C. S. Raghavendra, and K. M. Sivalingam, "Data gathering in sensor networks using the energy*delay metric", *Proceedings IPDPS'01*, San Francisco, CA, Apr. 2001, pp. 2001–2008.

[41] S. Lindsey, C. S. Raghavendra, and K. M. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics", *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 9, Sept. 2002, pp. 924–935.

[42] B. Krishnamachari, Y. Mourtada, and S. Wicker, "The energy-robustness tradeoff for routing in wireless sensor networks", *Proceedings IEEE ICC'03*, Seattle, WA, May 2003, pp. 1833–1837.

[43] S. Dulman, T. Nieberg, J. Wu, and P. Havinga, "Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks", *Proceedings IEEE WCNC'03*, New Orleans, LA, Mar. 2003, pp. 1918–1922.

[44] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks", *International Journal of High Performance Computing Applications*, vol. 16, no. 3, Fall 2002, pp. 293–313.

[45] X. Du and F. Lin, "Improving routing in sensor networks with heterogeneous sensor nodes", *Proceedings IEEE VTC'05*, Dallas, TX, Sept. 2005, pp. 2528–2532.

[46] Y.-B. Ko and N. H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks", *Wireless Networks*, vol. 6, no. 4, July 2000, pp. 307–321.

[47] B. Krishnamachari, D. Estrin, and S. Wicker, "Modeling data-centric routing in wireless sensor networks", *Proceedings IEEE INFOCOM'02*, New York, NY, June 2002, pp. 1–11.

[48] M. Stemm and R. H. Katz, "Measuring and reducing energy consumption of network interfaces in hand-held devices", *IEICE Transaction on Communications*, vol. E80-B, no. 8, Aug. 1997, pp. 1125–1131.

[49] W. Wang, V. Srinivasan, and K.-C. Chua, "Using mobile relays to prolong the lifetime of wireless sensor networks", *Proceedings ACM/IEEE MobiCom'05*, Cologne, Germany, Sept. 2005, pp. 270–283.

[50] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh, "Exploiting heterogeneity in sensor networks", *Proceedings IEEE INFOCOM'05*, vol. 2, Miami, FL, Mar. 2005, pp. 878–890.

[51] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Worst-case optimal and average-case efficient geometric ad-hoc routing", *Proceedings ACM MobiHoc'03*, Annapolis, MD, June 2003, pp. 267–278.

[52] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks", *Proceedings ACM MobiCom'00*, Boston, MA, Aug. 2000, pp. 243–254.

[53] T.-C. Hou and V. Li, "Transmission range control in multihop packet radio networks", *IEEE Transactions on Communications*, vol. 34, no. 1, Jan. 1986, pp. 38–44.

[54] I. Stojmenovic and X. Lin, "Geographic distance routing in ad hoc wireless networks", *Technical Report TR-98-10*, Computer Science Department, SITE, University of Ottawa, Canada, 1998.

[55] O. Kasten, "Energy Consumption", *www.inf.ethz.ch/~kasten/research/bathtub/energy_consumption.html*

[56] P. Bahl and V. N. Padmanabhan, "Radar: A in-building rf-based user location and tracking system", *Proceedings IEEE INFOCOM'00*, vol. 2, Tel-Aviv, Israel, Mar. 2000, pp. 775–784.

[57] L. Doherty, K. S. Pister, and L. E. Ghaoui, "Convex position estimation in wireless sensor networks", *Proceedings IEEE INFOCOM'01*, vol. 3, Anchorage, AK, Apr. 2001, pp. 1655–1663.

[58] M. Zorzi and R. R. Rao, "Geographic random forwarding (geraf) for ad hoc and sensor networks: Energy-Latency Performance", *IEEE Transactions on Mobile Computing*, vol. 2, no. 4, Oct.–Dec. 2003, pp. 349–365.

[59] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc., New York, 1991.

[60] W. Zhang, G. Cao, and T. La Porta, "Data dissemination with ring-based index for wireless sensor networks", *Proceedings IEEE ICNP'03*, Atlanta, GA, Nov. 2003, pp. 305–314.

[61] R. Kumar, V. Tsiatsis, and M. B. Srivastava, "Computation hierarchy for in-network processing", *Mobile Networks and Applications*, vol. 10, no. 4, Aug. 2005, pp. 505–518.

[62] S. Nath and P. B. Gibbons, "Communicating via fireflies: Geographic routing on duty-cycled sensors", *Proceedings IPSN'07*, Cambridge, MA, Apr. 2007, pp. 440–449.

[63] D. Pompili, T. Melodia, and I. F. Akyildiz, "Routing algorithms for delay-insensitive and delay-sensitive applications in underwater sensor networks", *Proceedings ACM MobiCom'06*, Los Angeles, CA, Sept. 2006, pp. 298–309.